Introduction

COMP 3700
Software Modeling and Design

This course is about…

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

What is Software?

- Program
- Categories
  - Personal / Limited-use software
  - Industrial-strength software

Personal / Limited-use software

- Limited set of behaviors
- Not very complex
- Developed by small group
  - May not be tech-savvy
- Short life span
- Can be thrown away
- No specific interest in development approach

- An example…

Industrial-strength software

- Rich set of behaviors
- Works with limited resources
- Maintains integrity of millions of records
- Commands and controls of real-world entities
- Long life span
- Depended by many users on proper functioning
- Usually based on frameworks
- Highly complex

Software is inherently complex

- Highly complex
  - Why?

Software is inherently complex

- Three contributing elements
  1. Complexity of problem domain
  2. Difficulty of managing development process
  3. Flexibility possible through software

1. Complexity of problem domain

- Domains are difficult to understand
  - Multi-engine aircraft systems
  - Merchant shipping
  - Online trading
- Complex functional requirements
- Unclear non-functional requirements
  - Often implicit
  - Difficulty to justify in budget

1. Complexity of problem domain (Contd.)

- Communication gap between users and developers
  - Leads to external complexity
- Evolving / Changing requirements
- Large investment
  - Need for software preservation

2. Difficulty of managing development process

- Large code bases
- Multiple teams
- Geo dispersion of groups
- Complex communication
- Difficult coordination

- Human intensive

3. Flexibility possible through software

- Build / buy components?
- Reusable components
  - Few standards exist
- Flexibility to change
  - Change in reqs. possible with software
  - Others: Not feasible.

Software is inherently complex: Review

- Three contributing elements
  1. Complexity of problem domain
  2. Difficulty of managing development process
  3. Flexibility possible through software

Software development / construction?

- Software is developed, not constructed.

- Domain evolution
  - Bridges
  - Surgery
  - Airplanes
  - Software

Software development / construction? (Contd.)

- Is software delivered successfully?
  - On time
  - Within budget
  - Complete & correct functionality
  - Without failures
- Software failures

https://spectrum.ieee.org/static/the-staggering-impact-of-it-systems-gone-wrong

- But why do software projects fail?

Why software projects fail?

- Complexity
  - Changes from requirements
  - Changes from technology
  - Changes from people

How to improve success rate?

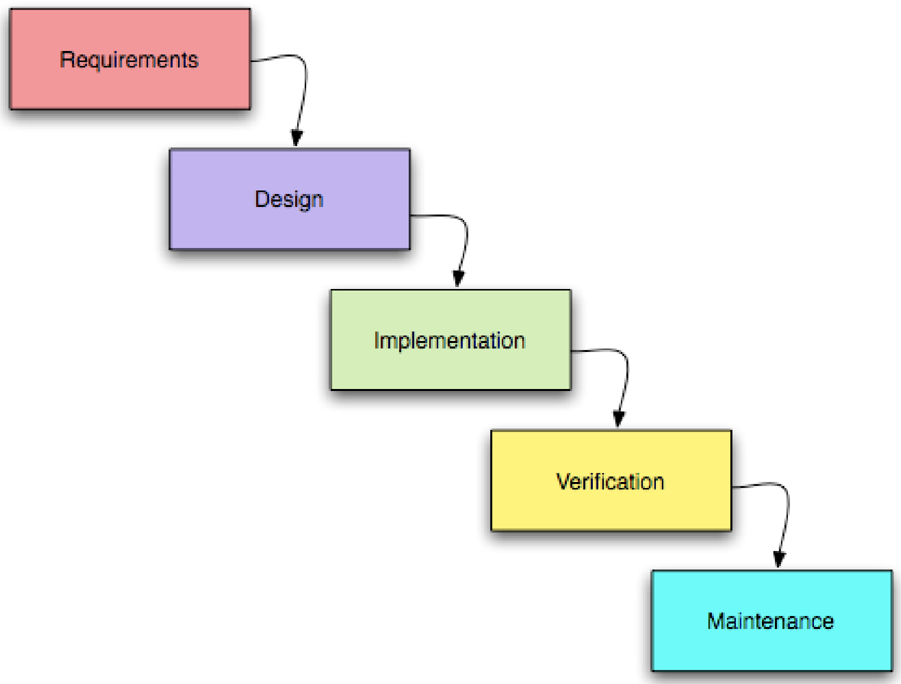
- Structured approach
- Adherence to best practices
- Reusing components

Software Development Approach

- Phases
  - Requirements
  - Design
  - Implementation
  - Validation
  - Maintenance

Software Development: Waterfall approach

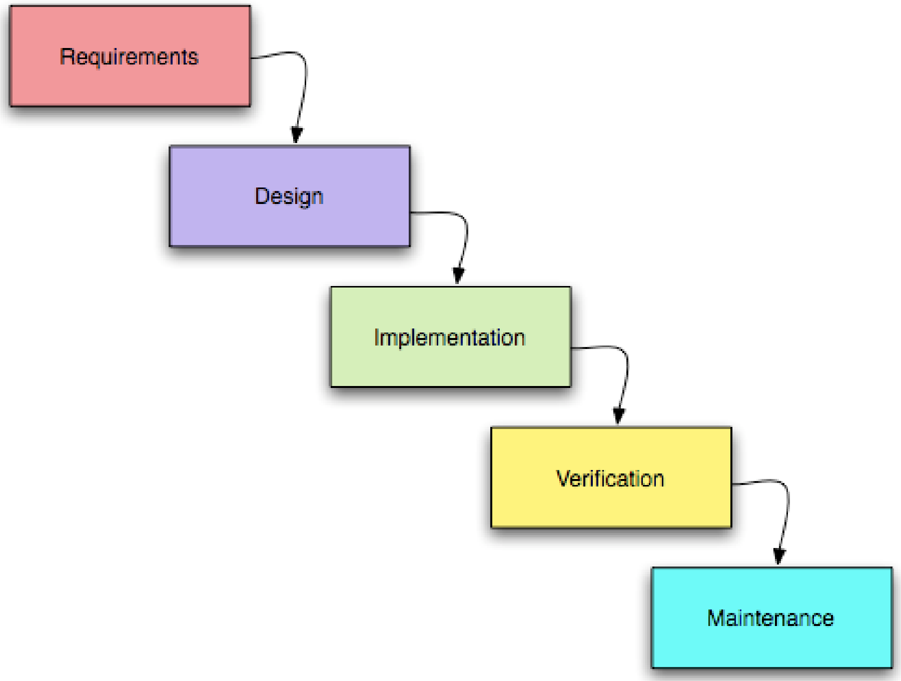
- Sequential approach
  - Strict linear sequence
  - No backtracking



Software Development: Waterfall approach

- Sequential approach
  - Strict linear sequence
  - No backtracking

- Does this suffice for all applications?

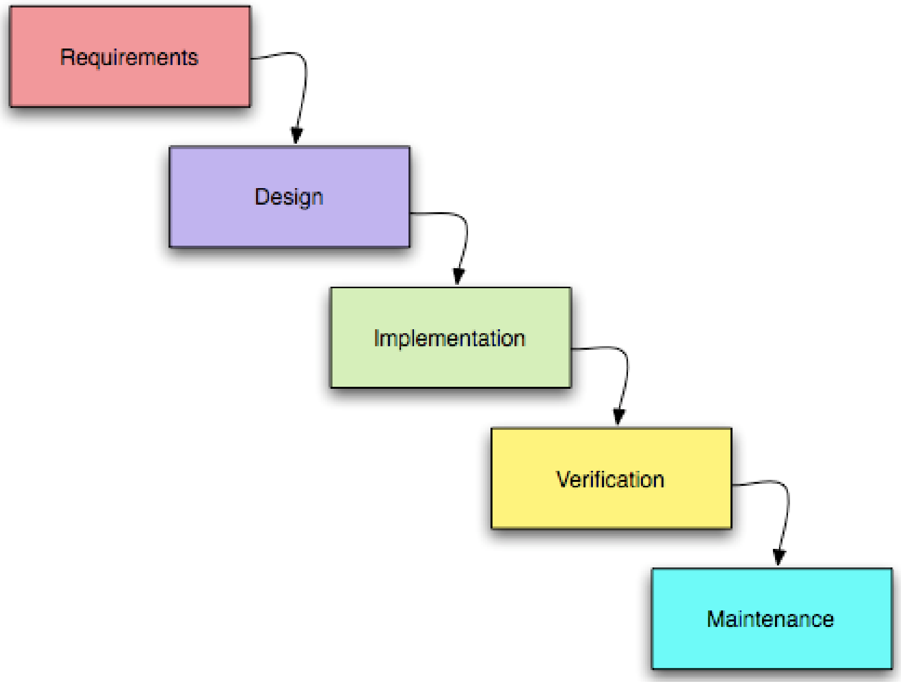


Software Development: Waterfall approach (Contd.)

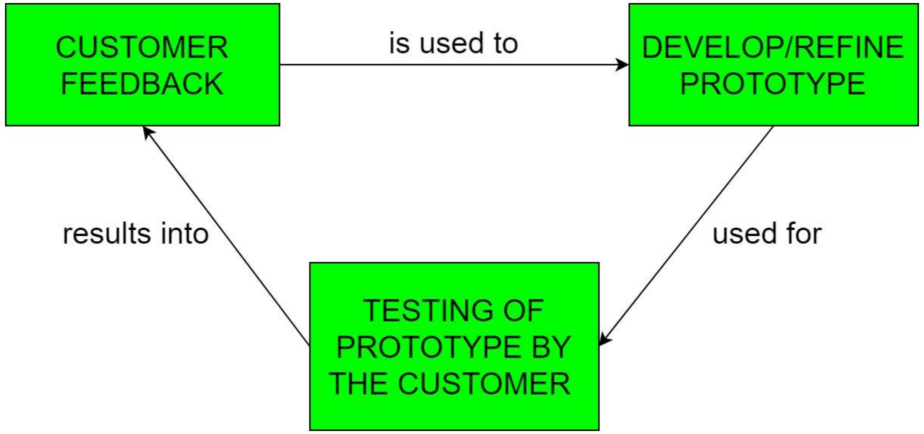
- Applicability
  - Well-understood and stable applications
- Limitations
  - Limited applicability
  - Does not deliver a useful system until completion
  - Difficult to assess progress
  - Difficult to correct project that drifted away from reqs
- High failure rate Why?

Waterfall approach (Contd.)

- How to overcome limitations?
- Any better approach?



Rapid Prototyping



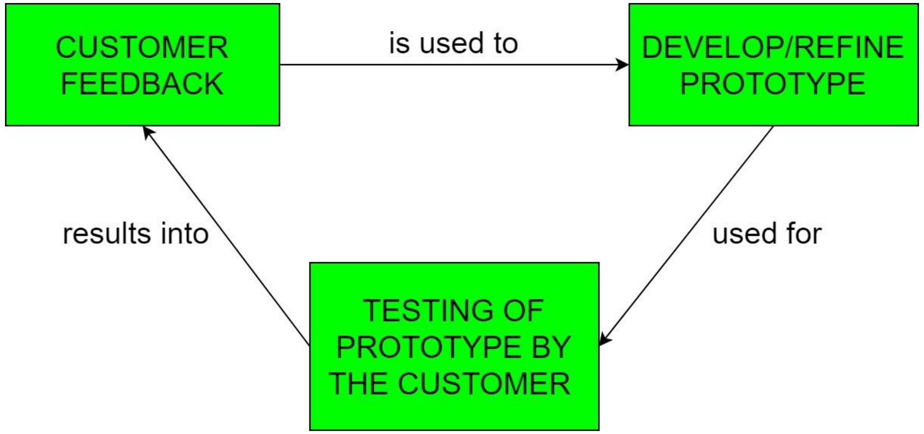
- Approach
  - Develop portion of software
  - Evaluate it
  - Receive user feedback

  - Repeat until satisfactory
  - Deliver final prototype as finished application
- Is this sufficient for all applications?

Rapid Prototyping (Contd.)

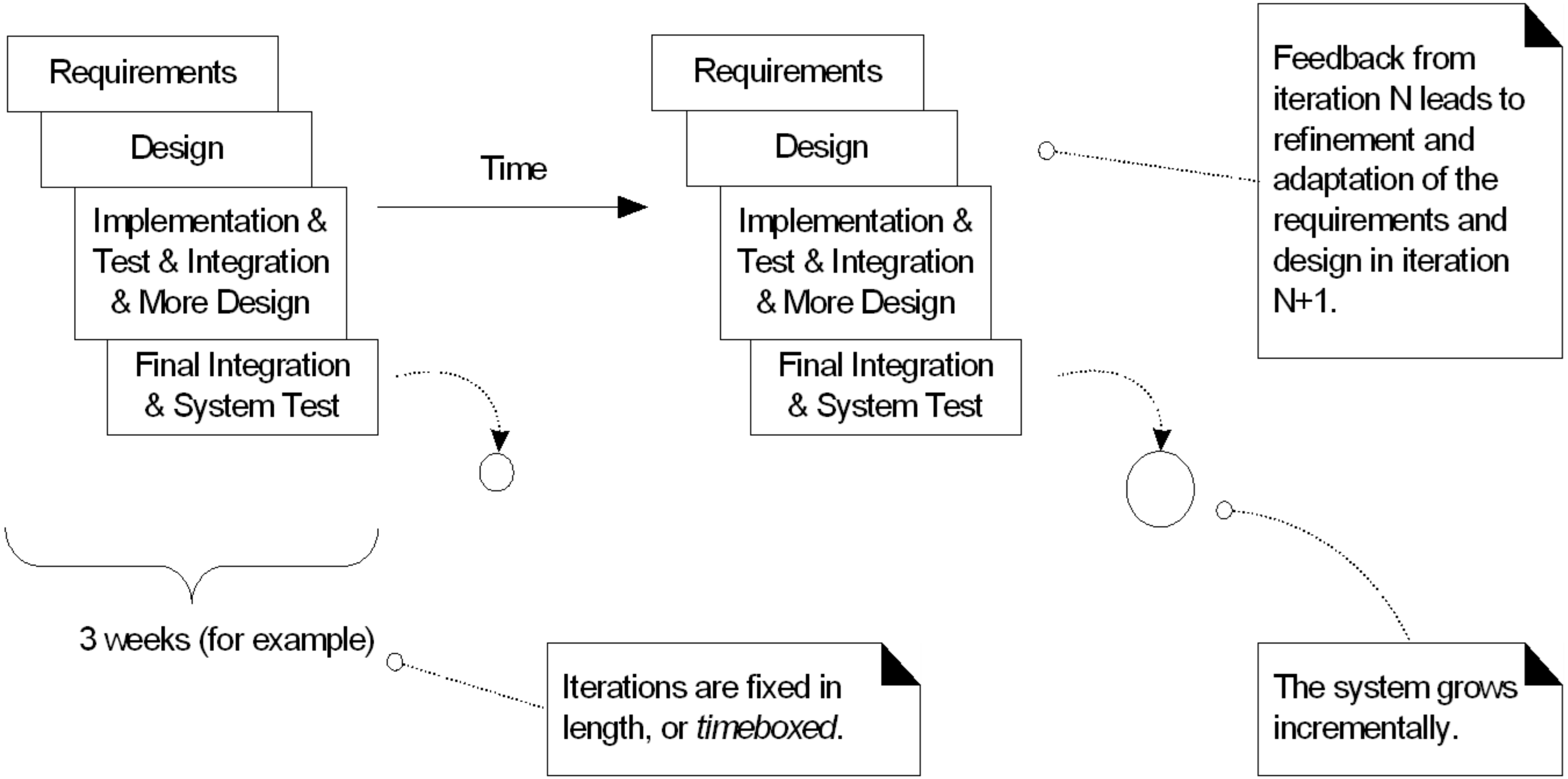
- Benefits
  - Promotes communication
    - Provides checkpoints for user validation and assurance
    - Resolve issues early
  - Helps elicit requirements
  - Demonstrate technical feasibility
- Drawbacks
  - Prototype is just a demonstration
    - May lack robust infrastructure
  - Difficult to discard code

Rapid Prototyping (Contd.)



- How to overcome limitations?
- Any better approach?

Iterative approach



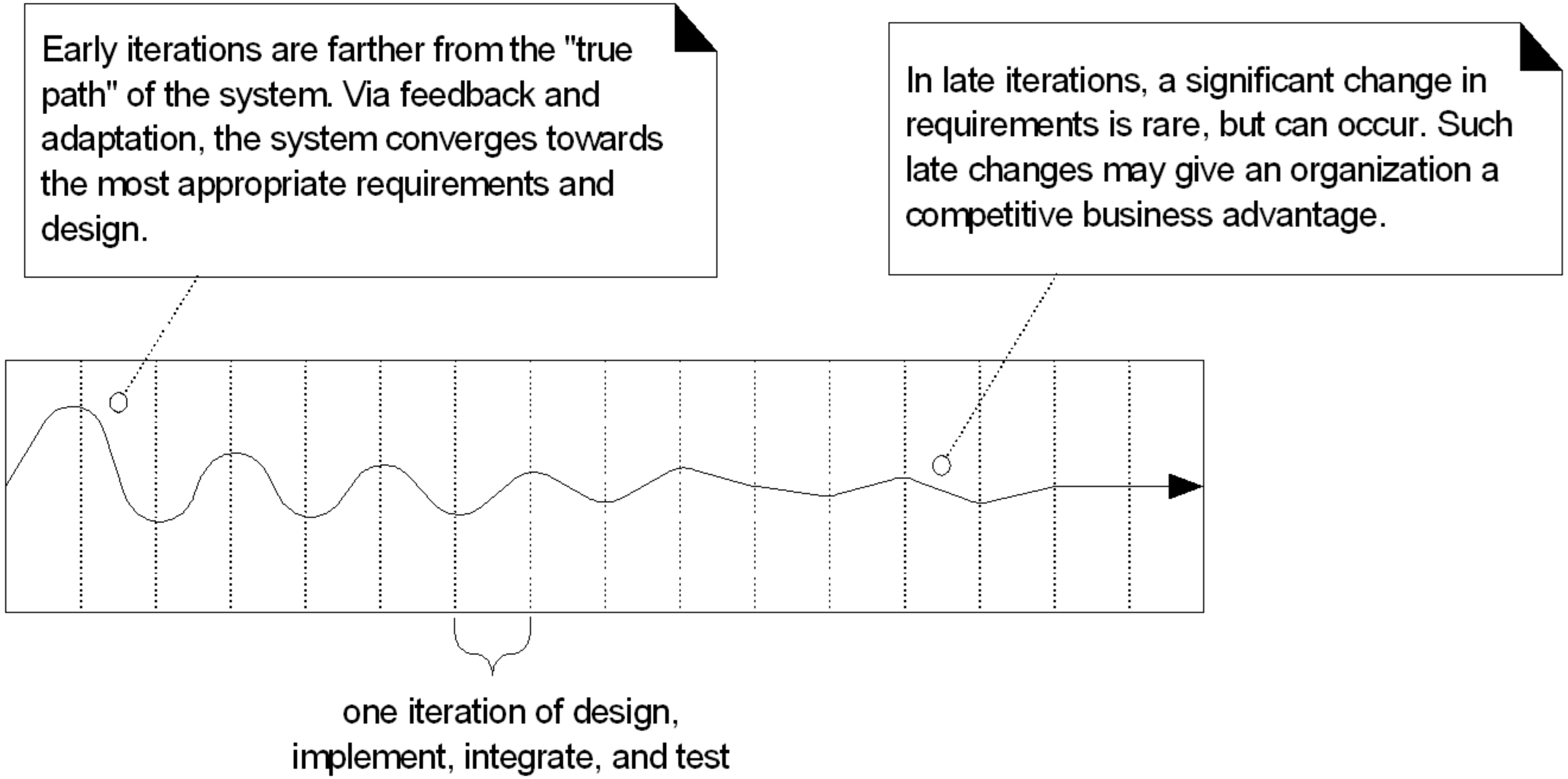
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
3 weeks (for example)

Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.

Iterations are fixed in length, or *timeboxed*.

The system grows incrementally.

Iterative approach



Early iterations are farther from the "true path" of the system. Via feedback and adaptation, the system converges towards the most appropriate requirements and design.

In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.

one iteration of design, implement, integrate, and test

Iterative approach: Planning

- Number of iterations
- Duration of each iteration
  - 2-6 weeks
  - Too small → high overhead
  - Too large → Insufficient checkpoints
  - Uniform length
- Iteration scope
  - Few use cases
    - Must have return value
  - Deliver executable code
- May combine iterations per release

Iteration planning

- Risk-aware
  - Technical risks
  - Technology risks
  - User acceptance risks
  - Schedule risks
  - Personnel risks
  - Market risks
- Evolutionary
- Adaptive

Discussed so far …

- Software development
  - Complexity
  - Reasons for failure
- Software development approaches
  - Waterfall approach
  - Rapid prototyping
  - Iterative approach

This course is about…

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

Why software design?

- High quality software
  - Complete
  - Correct
  - Efficient
  - Robust
  - Reusable
  - Modular
  - Easy to understand, update, and integrate
  - …

This course is about…

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

What is a model?

- An abstraction of something for the purpose of understanding it before building it.
  - Easier to manipulate
  - Testing a physical entity before building it
    - Cheaper to build
    - Provides fleeting / inaccessible metrics
  - Communication with customers
  - Visualization
  - Reduction of complexity (Human: 7±2 Pieces)

This course is about…

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

What is Object Oriented?

Compare with…

Procedure Oriented

What is Object Oriented?

- Organization of software as a collection of discrete objects that incorporate both data structure and behavior.
  - Represent
  - Behave
  - Interact

OO Characteristics

- Identity
  - Objects are discrete distinguishable entities
- Classification
  - Class includes objects with same data structure and behavior
- Inheritance
  - Sharing of attributes and operations among classes based on a hierarchical relationship
- Polymorphism
  - Same operation may behave differently for different classes in hierarchy.

OO Themes

- Abstraction
  - Focus on essential aspects of application while ignoring details
- Encapsulation (Information hiding)
  - Separates external aspects of object (accessible to other objects) from internal implementation details (hidden from other objects)
- Combining data and behavior
  - Operator polymorphism
- Sharing code
  - Inheritance of both data structure and behavior
  - Reusing designs and code on future projects
  - Build libraries of reusable components

OO Terms

- Object Oriented Analysis
- Object Oriented Design
- Object Oriented Programming
- Object Oriented Methodology

Object Oriented Analysis

A method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

Object Oriented Design

A method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical/physical as well as static/dynamic models of the system under design.

Object Oriented Programming

A method of implementation in which programs are organized as cooperative objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

Object Oriented Methodology

- Process for OO development.
- Stages:
  - System conception
  - Analysis
  - System design
  - Class design
  - Implementation

OO Models

- Class Model
  - Static structure of objects and relationships
    - Class diagram
- State Model
  - Changes over time or on events
    - State diagram
- Interaction Model
  - Interaction among objects
    - Use case diagram
    - Sequence diagram
    - Activity diagram

This course is about…

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

Visual Modeling using UML

- Unified Modeling Language
  - Standard graphical notation
  - Captures business processes
  - Communication tool
  - Manages complexity
  - Independent of platform / language
  - Facilitates documentation

Discussed so far …

A brief overview of

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

Next sessions…

- Class Modeling