Introduction

COMP 3700.002
Software Modeling and Design

Shehenaz Shaik

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
- Specified, constructed, maintained, and used by same person / small group
  - May / may not be tech-savvy
- Short life span
- Can be thrown away
  - Not reuse / repair / extend functionality
  - Minimal loss of investment
- No specific interest in development approach
- An example…

Industrial-strength software

- Rich set of behaviors
  - E.g. Event-based reactive systems in physical world
- Works with limited resources
- Maintains integrity of millions of records
  - Databases allowing concurrent updates and queries
  - E.g. Airline bookings
- Commands and controls of real-world entities
  - E.g. Air/Rail traffic routing
- Long life span
- Depended by many users on proper functioning
- Usually based on frameworks
  - Which simplify creation of domain-specific applications
- Highly complex

Software is inherently complex

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

1. Complexity of problem domain

- Domains are difficult to understand
  - Multi-engine aircraft systems
  - Merchant shipping
  - Online trading
- Functional requirements
  - Difficult to master
  - Often are competing, may be contradictory
- Non-functional requirements
  - Often implicit
  - Difficulty to justify in budget

1. Complexity of problem domain (Contd.)

- Communication gap between users and developers
  - Lack of expertise across domains
  - Different perspectives →Different solutions
  - Difficulty in precise capture of requirements
    - Text / Diagrams
  - Leads to external complexity
- Evolving / Changing requirements during development
  - Early versions lead to better understanding of needs by users
  - Developers understand the domain better
- Large investment
  - Difficult to discard, as requirements change
  - Results in software preservation
    - Maintenance Vs. Evolution Vs. 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?
  - Builds components often within team
  - Other industries e.g. civil: sources components
- Few standards exist for reusable components
  - Civil: Uniform building codes and established standards for raw materials
- Flexibility to change
  - Change in reqs. possible with software
  - Others: Not feasible.

Software is inherently complex: Review

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

Software development / construction?

- Software is developed, not constructed.
  - Solve a problem – Innovative
  - Created, which didn't exist before
- Comparison of domain evolution
  - Bridges / Surgery / Airplanes / Software
- Is software delivered successfully?
  - On time
  - Within budget
  - Complete & correct functionality
  - Without failures

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

Why projects fail?

- Complexity
  - Changes from requirements
    - Users / developers learn
    - Changes in user environment
  - Changes from technology
    - Hardware / Network / Platform
  - Changes from people
    - Complex interactions
    - Unpredictable behavior

How to improve success rate?

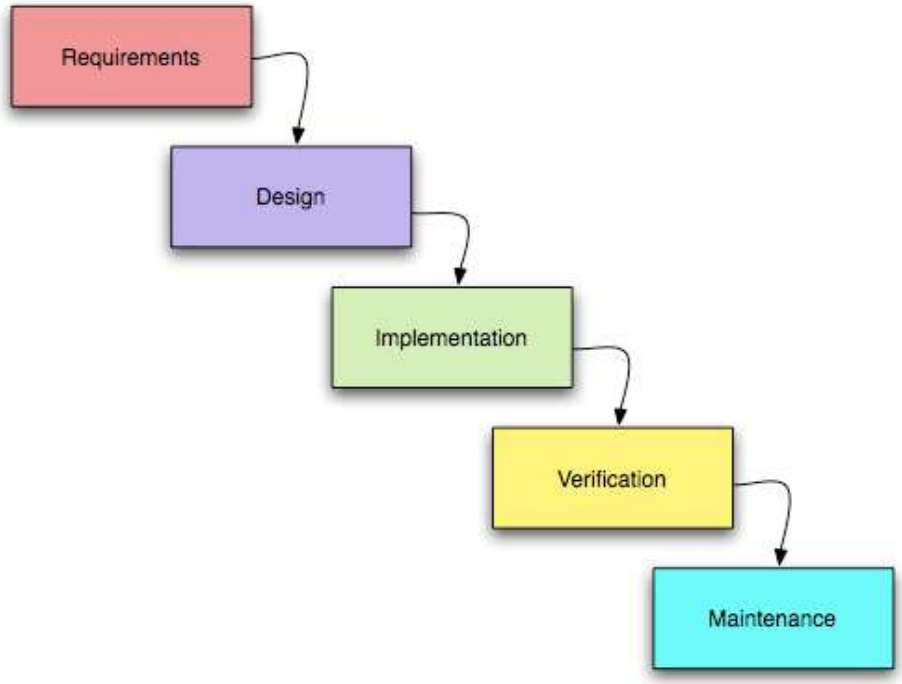
- Structured analysis, modeling, design, and implementation
- Adherence to best practices
- Reusing components

Software Development Lifecycle

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

Waterfall approach

- Sequential approach
  - Strict linear sequence
  - Each stage must complete prior to next
  - No backtracking
- Does this suffice for all applications?

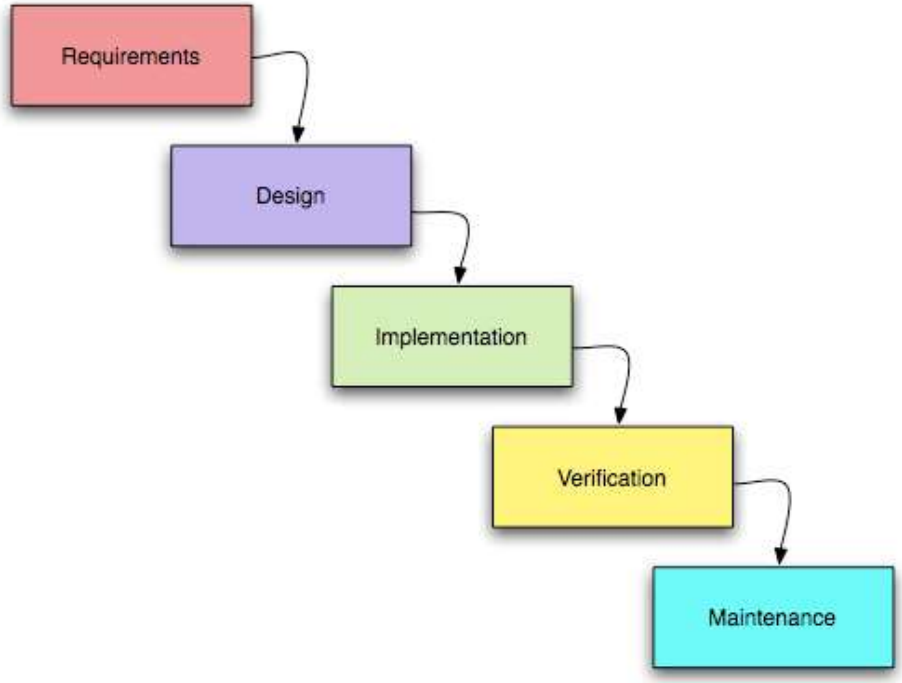


Waterfall approach (Contd.)

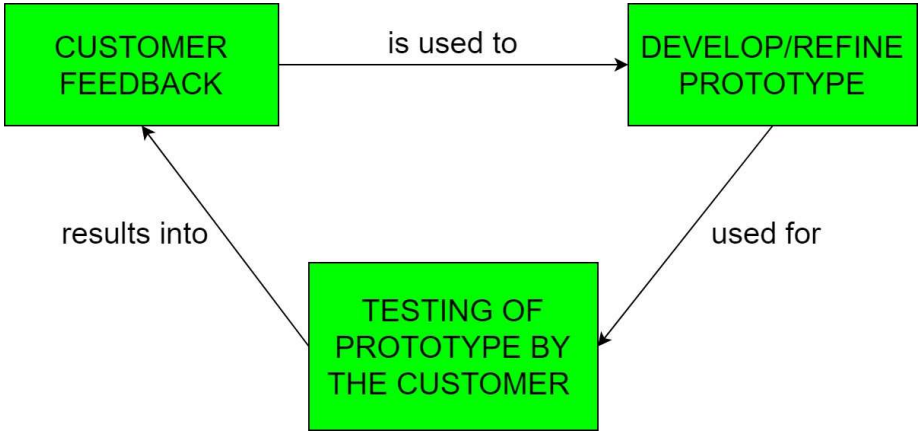
- Applicability
  - Well-understood applications
  - Predictable outputs from analysis and design
  - Clear and stable requirements
- Limitations
  - Limited applicability
  - Does not deliver a useful system until completion
  - Difficult to assess progress
  - Difficult to correct project that drifted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

Waterfall approach (Contd.)

- How to overcome limitations?
- Any better approach?



Rapid Prototyping



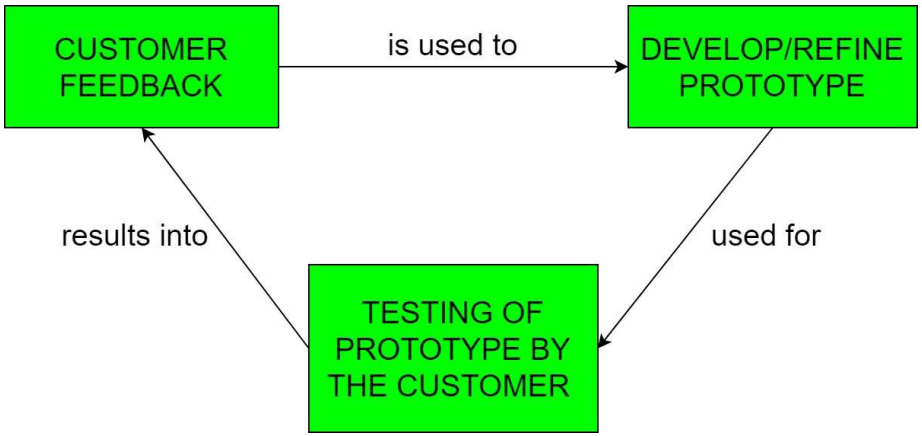
- Approach
  - Develop portion of software
  - Evaluate it
  - Receive user feedback

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

Rapid Prototyping (Contd.)

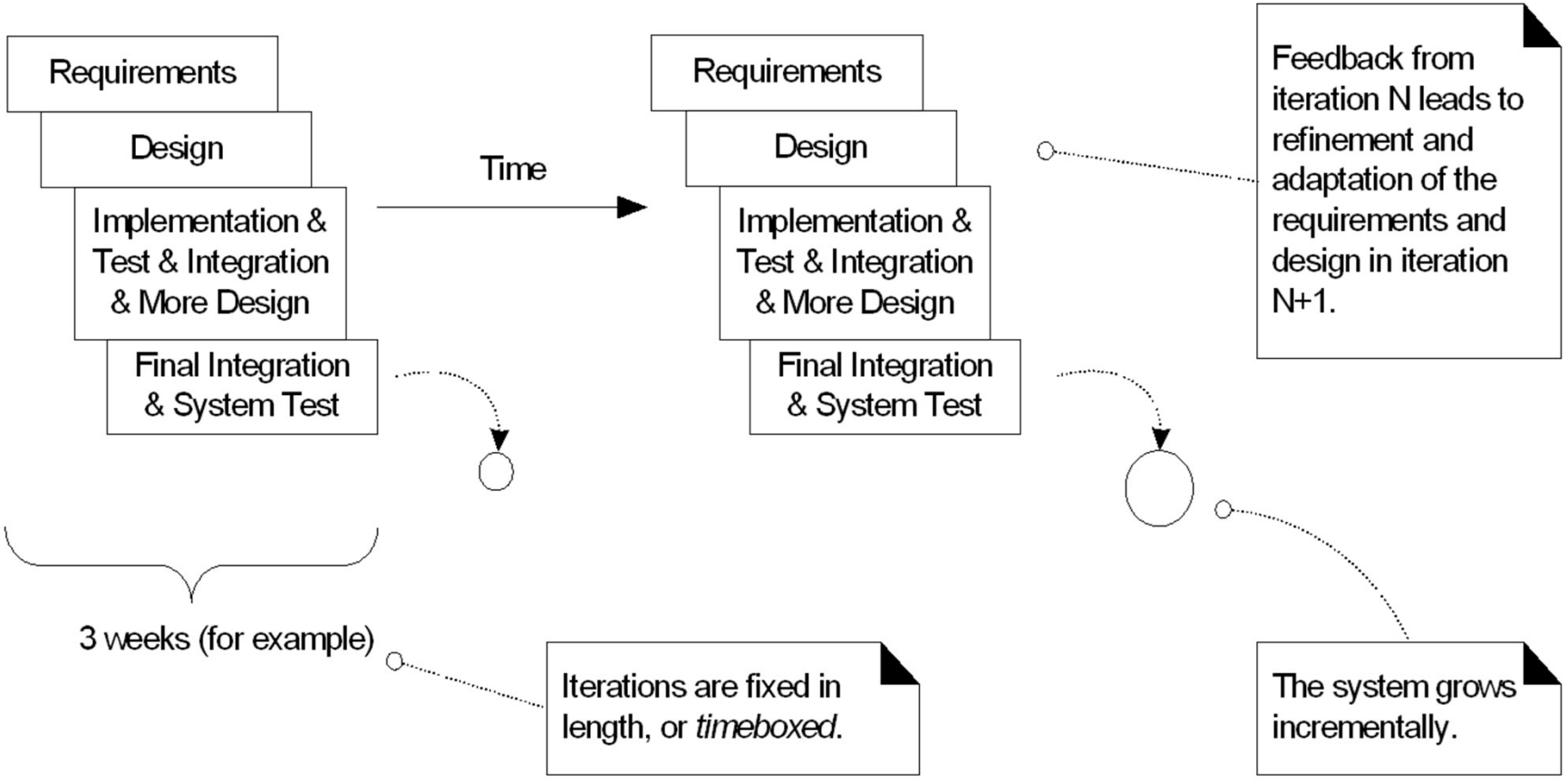
- Benefits
  - Promotes communication
    - Provides checkpoints for user validation and assurance
    - Resolve issues early
  - Helps elicit requirements
  - Demonstrate technical feasibility
- Drawbacks
  - Prototype is just a demonstration
    - May lack robust infrastructure
  - Difficult to discard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

Rapid Prototyping (Contd.)

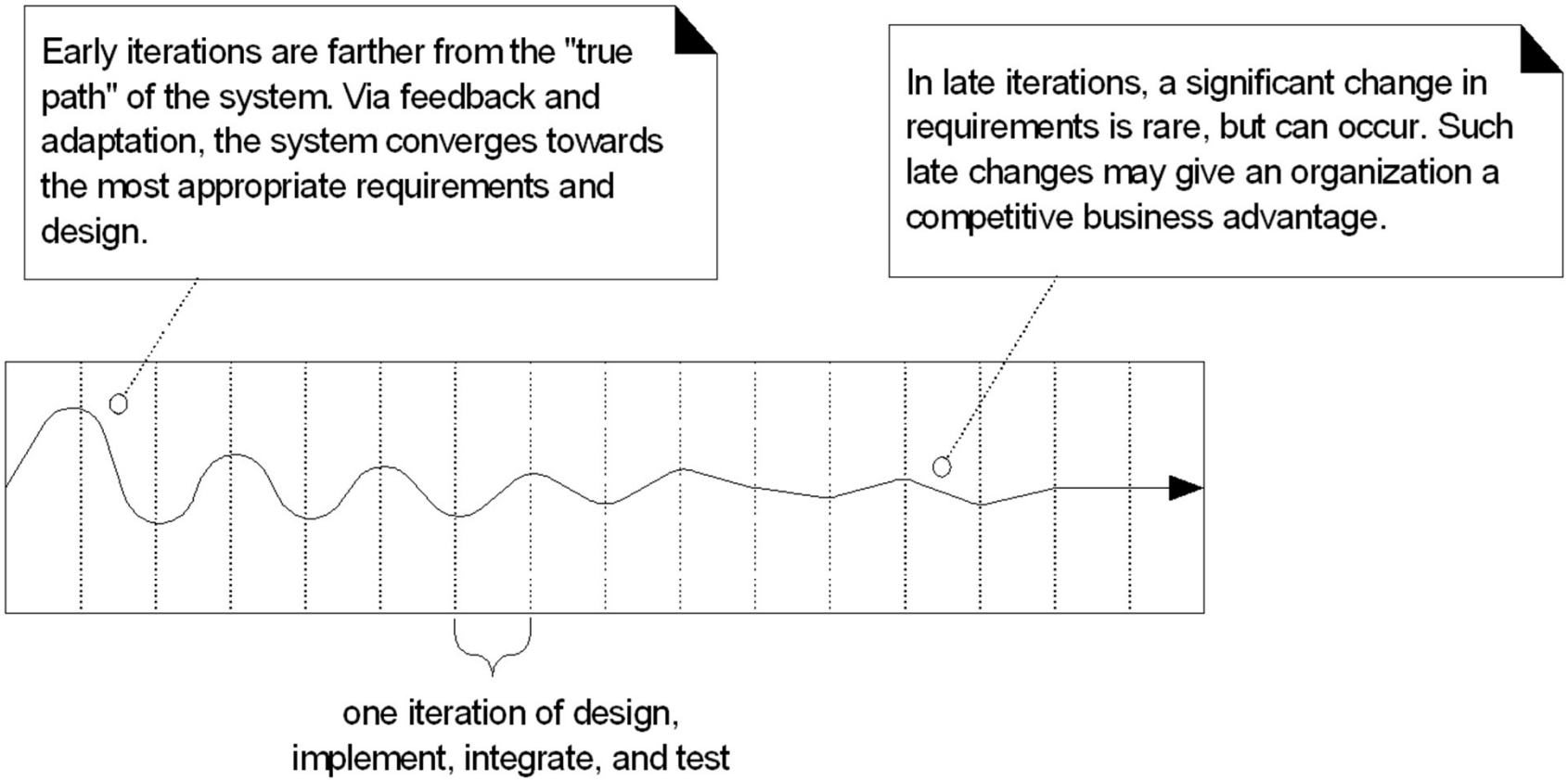


- How to overcome limitations?
- Any better approach?

Iterative approach



Iterative approach



Iterative approach

- 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
    - economic payback, added functionality, improved user interaction, better efficiency, higher reliability, or strengthened infrastructure for maintenance and future iterations
  - 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 Orientation?

- Organization of software as a collection of discrete objects that incorporate both data structure and behavior.
  - What do objects represent?
  - How do objects behave?
  - How do objects 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 and physical as well as static and 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
    - Concise, precise abstraction
    - Domain model
    - Application model
  - System design
    - System architecture, Interactions
    - Establish policies
  - Class design
    - Interfaces, Data structures and Algorithms
  - 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

Reading assignment

- [Blaha] Chp 21: Pages 395-401
- [Larman] Chp 2: Pages 17-40
- [Blaha] Chp 1: Pages 1-10
- [Blaha] Chp 2: Pages 15-18