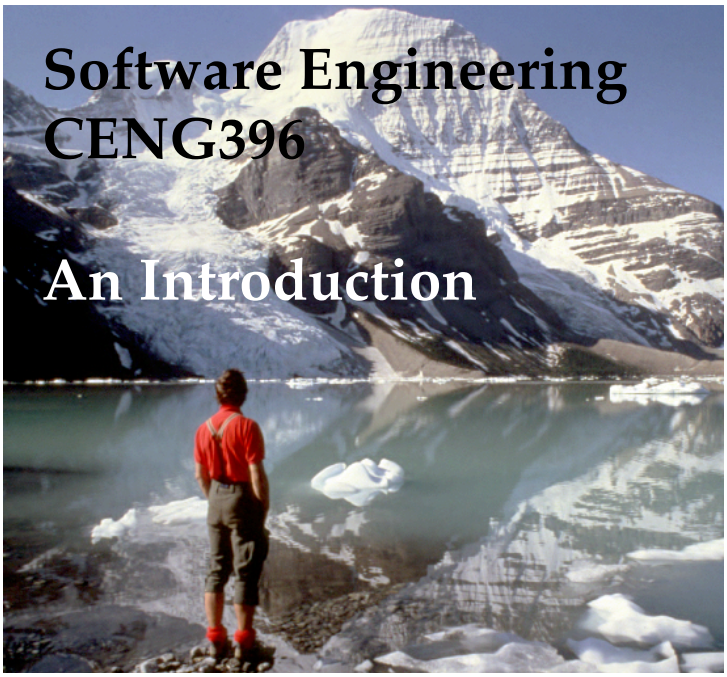
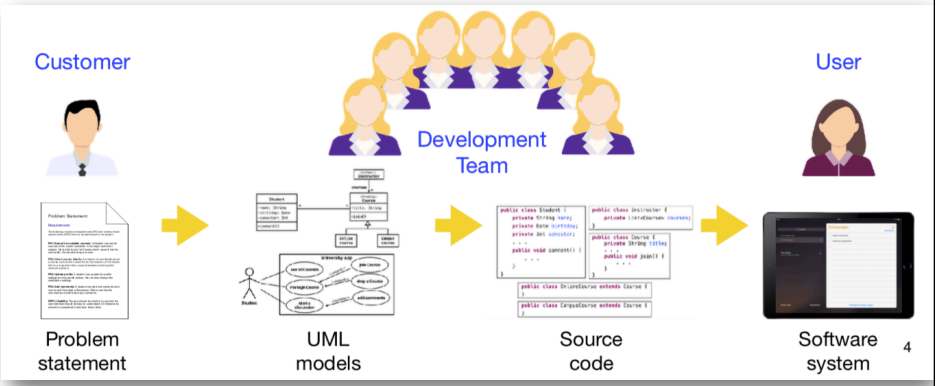
Object-Oriented Software Engineering
Using UML, Patterns, and Java



Software Engineering – What is?



- Learn and apply methodologies, techniques, workflows and tools to transform a problem statement given by the customer into a software system used by end users

2

Motivation for Software Engineering

- Complexity
- Innovation
- Flexibility
- Recognition
- Opportunities

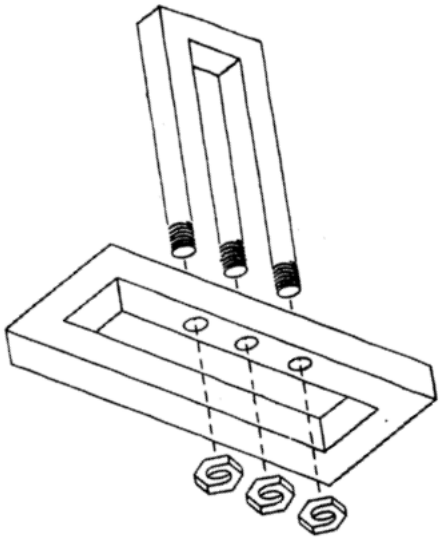
3

Outline of Today's Lecture

- The development challenge
- Dealing with change
- Concepts: Abstraction, Modeling, Hierarchy
- Methodologies
- Course syllabus

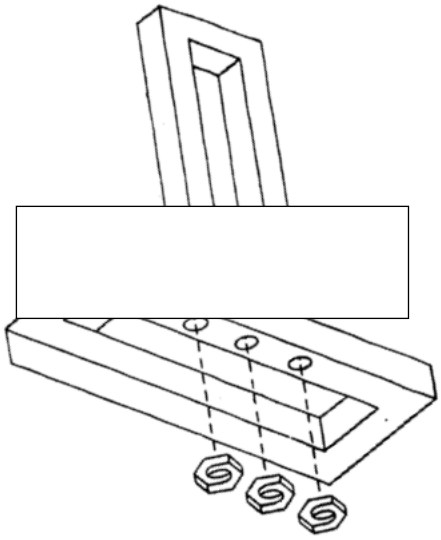
4

Can you develop this system?



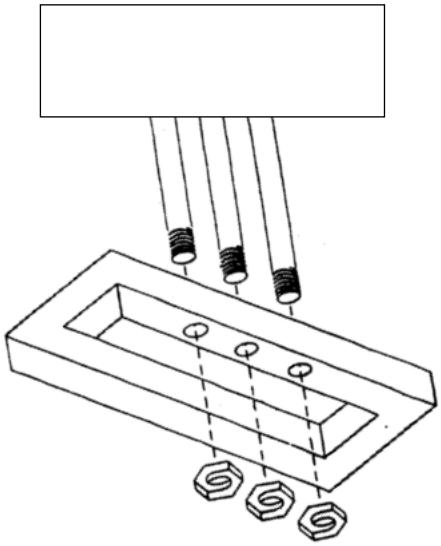
5

Can you develop this system?



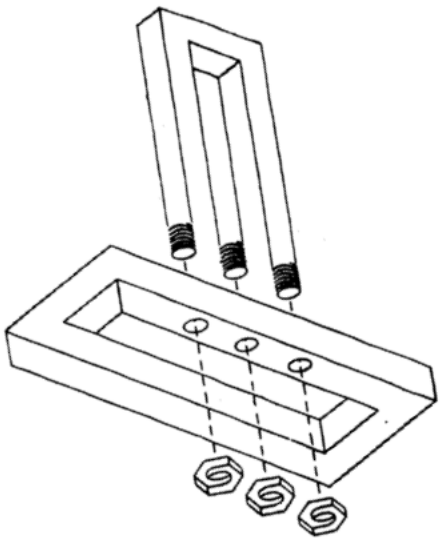
6

Can you develop this system?



7

Can you develop this system?



The impossible Fork

8

Physical Model of the Impossible Fork (Shigeo Fukuda)

https://www.youtube.com/watch?v=1XfbiziIHmk&feature=youtu.be

Escher: Mathematical Art
https://www.youtube.com/watch?v=Kcc56fRtrKU

9

10

Why is software development difficult?

- The **problem domain** (also called application domain) is difficult
- The **solution domain** is also difficult
- The **development process** is difficult to manage
- Software offers extreme flexibility
- Software is a discrete system
  - Continuous systems have no hidden surprises
  - Discrete systems can have hidden surprises! (Parnas)

David Lorge Parnas is an early pioneer in software engineering who developed the concepts of **modularity** and **information hiding** in systems which are the foundation of object oriented methodologies.

11

Software Engineering is more than writing Code

- Problem solving
  - Creating a solution
  - Engineering a system based on the solution
- Modeling
- Knowledge acquisition
- Rationale management

12

Techniques, Methodologies and Tools

- **Techniques**
  - Formal procedures for producing results using some well-defined notation
- **Methodologies**
  - Collection of techniques applied across software development and unified by a philosophical approach
- **Tools**
  - Instruments or automated systems to accomplish a technique
  - CASE = Computer Aided Software Engineering

13

Computer Science vs. Engineering

- **Computer Scientist**
  - Assumes techniques and tools have to be developed.
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
  - Has infinite time…
- **Engineer**
  - Develops a solution for a problem formulated by a client
  - Uses computers & languages, techniques and tools
- **Software Engineer**
  - Works in multiple application domains
  - Has only 3 months...
  - …while changes occurs in the problem formulation (requirements) and also in the available technology.

14

Software Engineering: A Working Definition

- Software Engineering is a collection of techniques, methodologies and tools that help with the production of

  a **high quality** software system developed with a given **budget** before a given **deadline** while **change** occurs

Challenge: Dealing with complexity and change

Software Engineering: A Problem Solving Activity

- **Analysis**
  - Understand the nature of the problem and **break** the problem into pieces
- **Synthesis**
  - **Put** the pieces **together** into a large structure
- For problem solving we use techniques, methodologies and tools.

Focus: Acquire Technical Knowledge

- Different methodologies ("philosophies") to model and develop software systems
- Different modeling notations
- Different modeling methods
- Different software lifecycle models (empirical control models, defined control models)
- Different testing techniques (eg. vertical testing, horizontal testing)
- Rationale Management
- Release and Configuration Management

17

Acquire Managerial Knowledge

- Learn the basics of software project management
- Understand how to manage with a software lifecycle
- Be able to capture software development knowledge (Rationale Management)
- Manage change: Configuration Management
- Learn the basic methodologies
  - Traditional software development
  - Agile methods

18

Focus

Dealing with **Complexity**

- Notations (UML, OCL)
- Requirements Engineering, Analysis and Design
  - OOSE, SA/SD, scenario-based design, formal specifications
- Testing
  - Vertical and horizontal testing

Dealing with **Change**

- Rationale Management
  - Knowledge Management
- Release Management
  - Big Bang vs Continuous Integration
- Software Life Cycle
  - Linear models
  - Iterative models
  - Activity-vs Entity-based views

19

Software Engineering – Course Description

- The course introduces the **basic concepts** of software engineering, and **modern tools** and well-known software development **methodologies** e.g., waterfall technique, iterative development, agile processes. The goal is to train students **design and document all phases of software development life-cycle** starting from gathering software requirements, software design methodologies, and software testing. Topics such as software quality assurance, **project management** are also discussed.

20

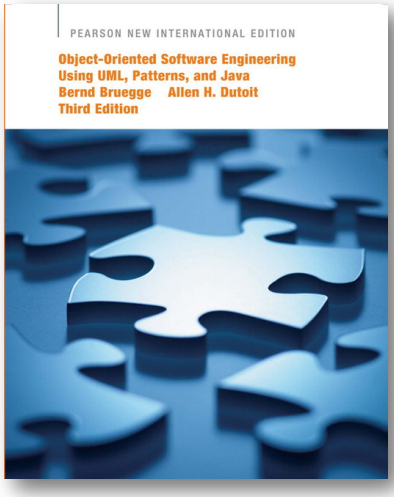
Objectives and Learning outcomes

- Learn the basics of the software engineering (SE) **process life cycle**.
- Learn what the **object-oriented (OO) approach to software development** is, through OO principles and design patterns.
- Learn **UML** (Unified Modeling Language) that is part of most **CASE** (Computer Aided Software Engineering) tools and the benefits of **visual modeling / diagramming**.
- Practice the application of principles of object-oriented software development through the course **group project**.
- Develop **teamwork** and **communication skills** through the course group project.

21

Textbook

- Object-Oriented Software Engineering Using UML, Patterns, and Java
  Bernd Bruegge
  Allen H. Dutoit
  Pearson, (3rd Ed.)
  ISBN 10: 1-292-02401-1



22

Weekly Lectures *(subject to change)*

- Introduction to SE (Ch 1)
- Modeling w/ UML (Ch 2)
- Project Organization and Communication (Ch 3)
- Requirements Elicitation (Ch 4)
- Analysis (Ch 5)
- System Design (Ch 6 & 7)
- Object Design (Ch 8 & 9)
- Mapping Models to Code (Ch 10)
- Testing (Ch 11)
- Rational Management (Ch 12)
- Configuration Management (Ch 13)
- Project Management (Ch 14)
- Project Life Cycle (Ch 15)

23

Lectures

- **Instructor**
  - Prof. Dr. Erdogan Dogdu
  - Computer Eng. Dept.
  - Email: edogdu@cankaya.edu.tr
  - Web: http://edogdu.cankaya.edu.tr
- **Office hours**
  - Wed 13:30-15:00
- **Lecture hours**
  - 3 hours/week
  - **Sec1:** Wed 09:20-11:10 (L111) and Thu 09:20-10:10 (L111)
  - **Sec2**: Wed 15:20-17:10 (L111) and Thu 10:20-11:10 (L111)

24

Objectives of the Lectures

- Appreciate the Fundamentals of Software Engineering:
  - Methodologies
  - Process models
  - Description and modeling techniques
  - System analysis - Requirements engineering
  - System design
  - Implementation: Principles of system development

25

Assumptions for this Class

- You have taken Object-Oriented (OO) Programming (**CENG241**) course and passed
- You know OO principles and an OO programming language well, such as **Java**, **C++**, or **Python**

26

Grading

- Attendance/Quiz/Assignments 15%
- Midterm 15%
- Project 40%
- Final Exam 30%

27

Project

- Develop a working software application
- Team work (4 students each)
- Agile software development methodology
- Tools: **GitHub** and **UML**
- Demo and presentation

28

Course Communication

- Web site
  - https://piazza.com/cankaya.edu.tr/spring2019/ceng396/home
  - The lecture slides will be posted in PDF format after the lecture is given
- Email
  - Do you have an **email client** installed on your phone?
  - Do you get email notifications?
  - Make sure you get course updates regularly and on time, all the time ☺

29

The End

- Assignment: Read Ch.1 from the textbook

30