Object Oriented Software Engineering

Lecture 1

Introduction to Software Engineering: Modelling

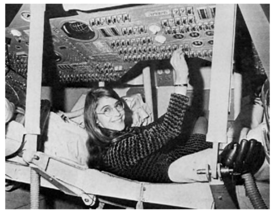
Dr. Graham McDonald
graham.mcdonald@glasgow.ac.uk
Room 406, SAW Building

Apollo 11

The spaceflight that landed the first two people on the Moon.
Commander Neil Armstrong and Lunar Module Pilot Buzz Aldrin.

Apollo 11

Margaret Hamilton: The woman whose code safely put humans on the moon.



Margaret Hamilton is credited with coining the term *software engineering*.

Software Engineering

Designing software then was not easy:

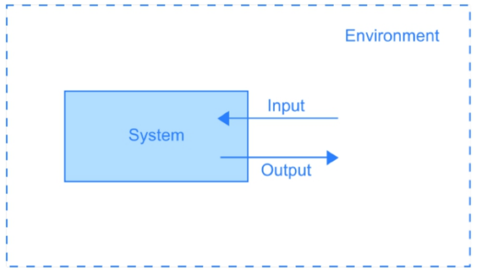
"When the computer crashed, sirens were loud... we had found a new way to debug, using sound."

Margaret Hamilton (ICSE 2018)

https://www.smithsonianmag.com/smithsonian-institution/margaret-hamilton-led-nasa-software-team-landed-astronauts-moon-180971575/

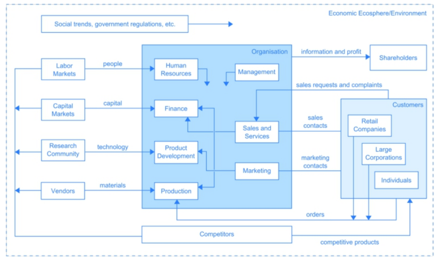
Understanding Systems

A simple system interacting with its environment using input and output messages.



Understanding Systems

A complex system with multiple sub-systems and interactions.



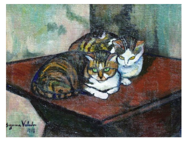
➢A software system is a web of interconnected sub-systems, where each sub-systems may or may not be divided into further sub-systems.

Abstraction

One way of understanding such complex systems in order to implement them is via abstraction.

Abstraction

Suppose an artist, a novelist and a programmer were asked to abstract (i.e., represent) a real-life object of an animal?



ANIMAL FARM
GEORGE ORWELL

Object Oriented Design

- An object combines data and operations on that data (object is an instance of class)
  - ➢ **Data**: class variables
  - ➢ **Operations**: methods

Object Oriented Design

- Three principles of Object Oriented Design

➢ **Encapsulation**:
Combining data and operations in one entity.

➢ **Inheritance**:
Classes can inherit from other classes (sub-classing) e.g. animals, dogs and so on.

➢ **Polymorphism**:
Means "many forms".

Object Oriented Software Engineering

Lecture 1: Part 2

Introduction to Software Engineering: Modelling

Dr. Graham McDonald
graham.mcdonald@glasgow.ac.uk
Room 406, SAW Building

Object Oriented Design

Given a problem statement and requirements, you carry out the following activities:

- ➢Identify Objects
- ➢Identify Operations
- ➢Create Interfaces
- ➢Object Interaction Design
- ➢etc.

Object Design – Identify Objects

- Identify (real-world) Objects:

  ➢Identify objects that exist in the problem Statement and requirements

  ➢Typically, select nouns, ignoring irrelevant ones, such as synonyms

Object Design – Identify Objects

- Look for relationships amongst the objects that were Identified

  ➢**Generalization** – relates to inheritance

  ➢**Containment** – where one object contains another

  ➢**Multiplicity** – determine the quantity relationships between objects
  (e.g. one animal can have many legs)

Object Design – Identify Operations

- Identify the operations of objects:

  ➢Typically selecting verbs from the problem statement

  ➢Associate each operation with the object that is responsible for providing the behaviour

Object Design – Create Interface

- An interface is created for each object that is to be represented by a class

  ➢The interface describes how the class can be used, by specifying its **public** operations.

- An interface should include:
  ➢Return type
  ➢Purpose (i.e. a description)
  ➢Pre and post conditions

Object Interaction Design

- Describe how the objects **communicate** with each other via **operations**.
- and how the object, operations and communication affects the end-users.

Software Design

Designing software is a **symbiotic relationship** between the end-user and designer that requires the software designer to make the right design decisions:

- Every design decision reflects an intent on how the software is to function or be used
- as well as end users’ expectations as to how the software is compatible with contextual norms.

What Makes A Good Software Design?

- A balance of:
  1. Modularity
  2. Modifiability
  3. Ease of Use
  4. Efficient
  5. Correct
  6. Maintainability
  7. Understandability
  8. Reusable
  9. Portable
  10. Fail-Safe
  11. ... etc