Introduction to Unified Modeling Language

Unified Modeling Language

- UML is an object-oriented modeling language (or more precisely, a collection of modeling languages) that is
  - expressive
  - semi-formal (UML 2.0 added much more formality)
  - capable of supporting incremental development
    - Elements can be hidden
    - Certain elements can be left incomplete
    - Inconsistencies can exist
  - process independent
    - UML can be used with a variety software development process models
  - Customizable and extensible

A Brief Timeline for OO and UML

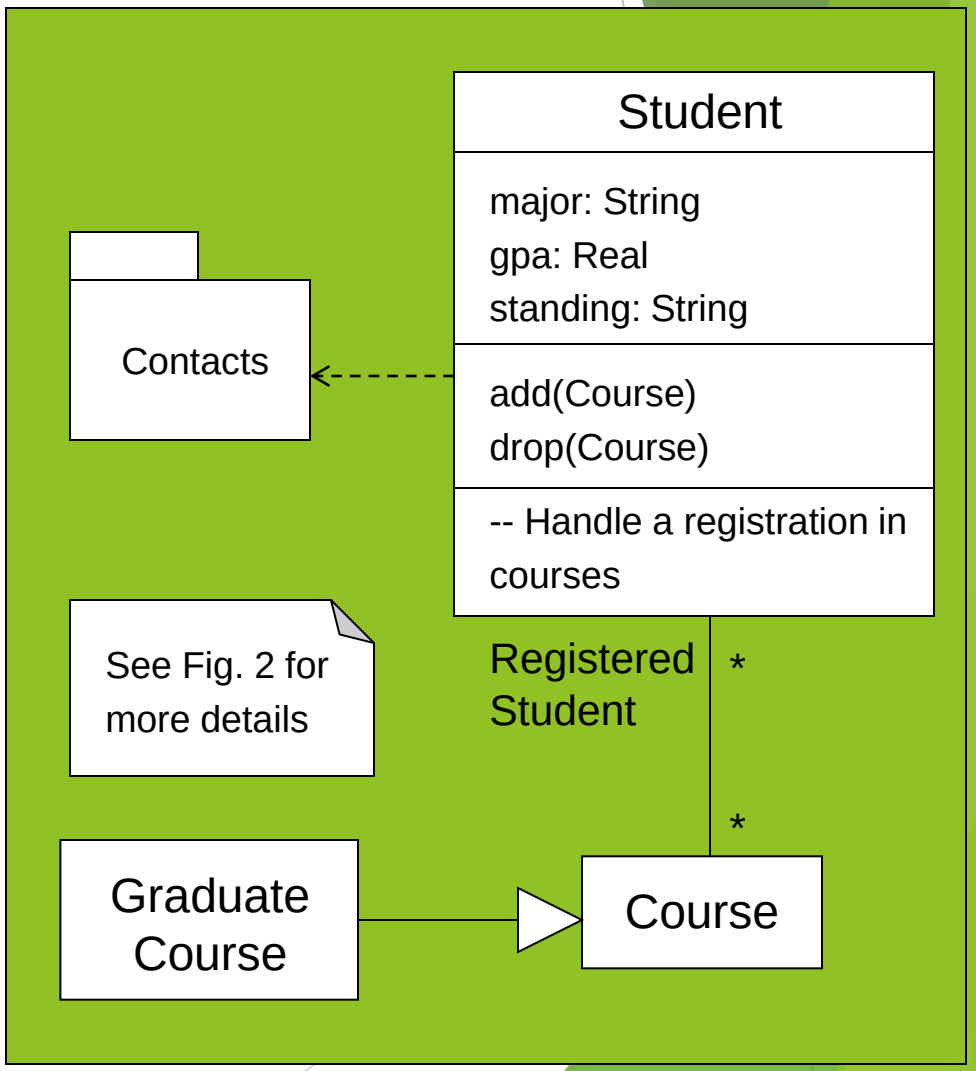
- 60's
  - Birth of initial OO ideas
- 70's
  - Nurturing of OO ideas
  - Introduction of a few more OO Programming Languages (OOPLs)
- 80's
  - Maturing of fundamental OO concepts
  - Emergence of more OOPL's
  - OOPL's gain widespread use

A Brief Timeline for OO and UML

- 90's
  - The Method Wars
  - Efforts to unify concepts
  - Introduction and standardization of UML
  - Emergence of next-generation ideas, like Patterns
- Current
  - Widespread use of UML
  - Widespread use Full-Life-Cycle development tools

UML Building Blocks

- Modeling Elements
  - Structural
  - Behavioral
  - Organizational
  - Annotational
- Diagrams that communicate ideas using the modeling elements
- Views



Modeling Elements

Modeling Elements are building blocks for constructing conceptual descriptions of systems

- Definition and Scope
  - Use Cases
  - Automation Boundaries
- Structural
  - Objects
  - Classes
  - Relations
  - Interfaces
  - Components
  - Nodes
- Extension
  - Templates
  - Stereotypes

- Behavioral Things
  - Messages
  - States
  - Transitions
  - Events
- Organizational Things
  - Packages
  - Views
- Annotation
  - Comments
  - Specifications

Modeling Elements and Diagrams

- Diagrams represent chunks of information that need to be communicated as part of a conceptual description.
  - It usually requires many diagrams to describe a system
  - Each diagram should focus on a single thought or a small set of tightly related thoughts
- Diagrams are like paragraphs in a section of well-structured text

UML Diagrams

- Use case diagrams
- Class Diagrams
- Object Diagrams
- Interaction Diagrams
  - Sequence Diagrams
  - Communication Diagrams
- State Charts (enhanced State Machines)
- Component Diagrams
- Deployment Diagrams

Classification of UML Diagrams

- Structure diagrams
- Behavior diagrams
- Interaction diagrams

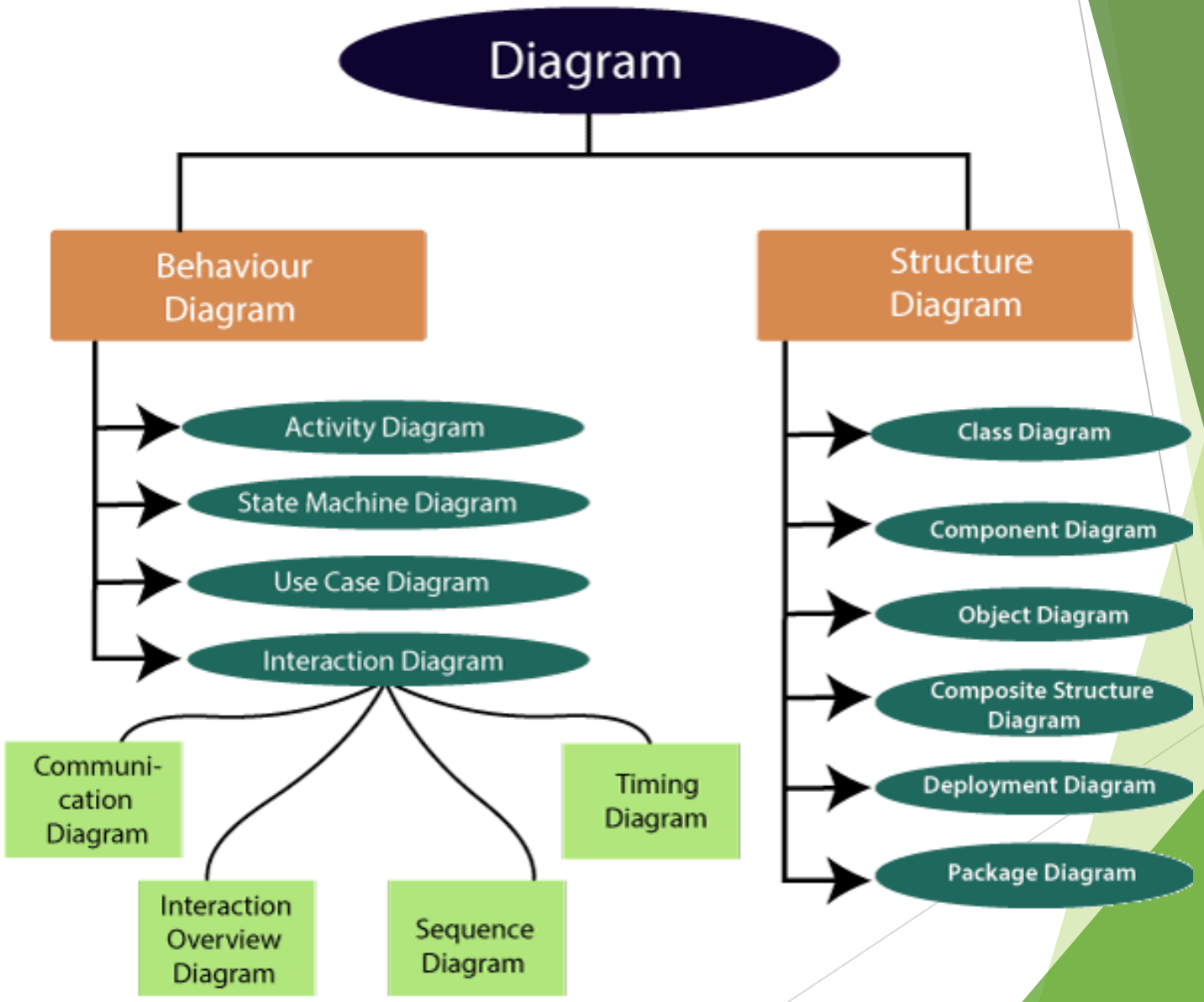
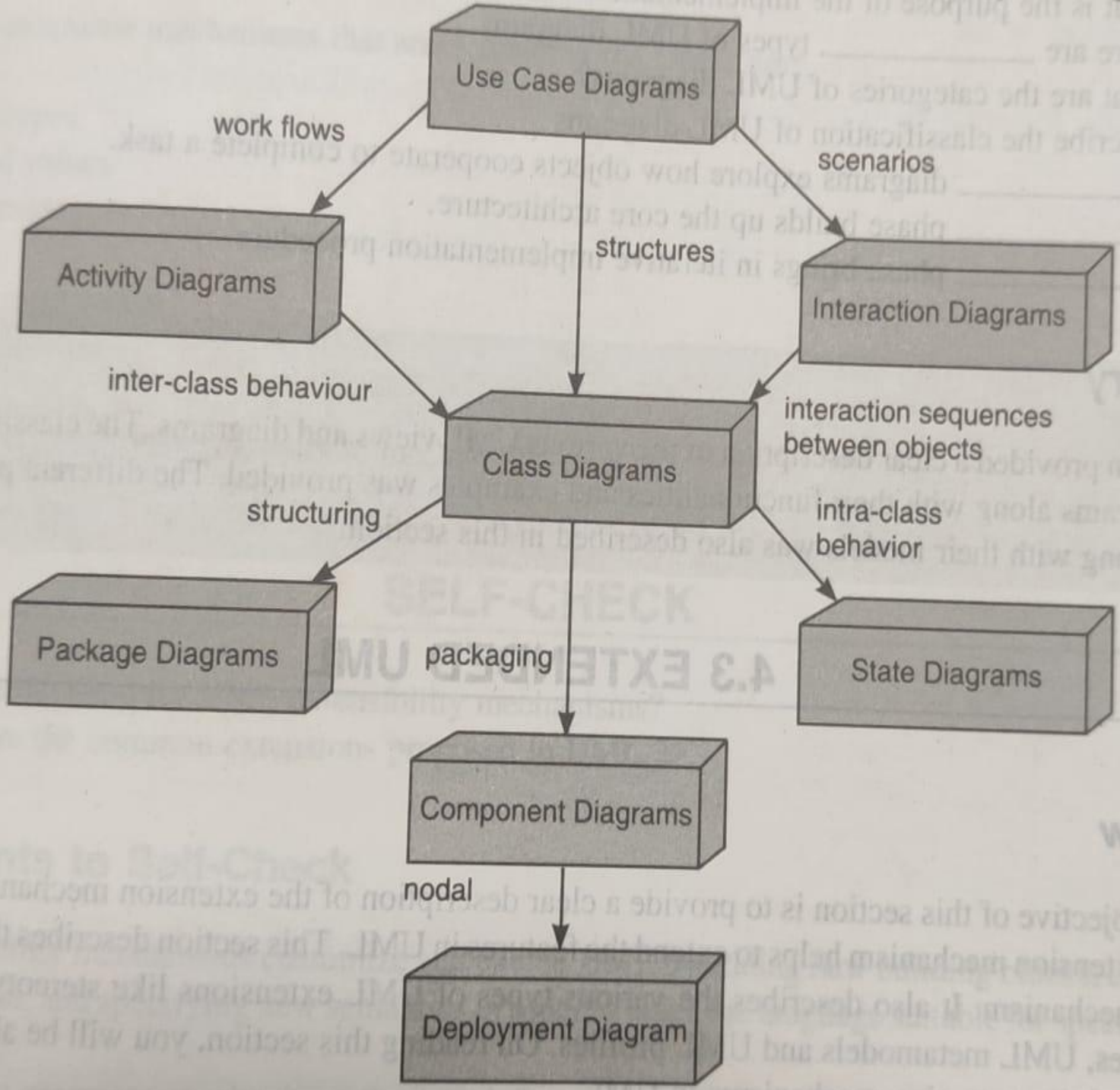


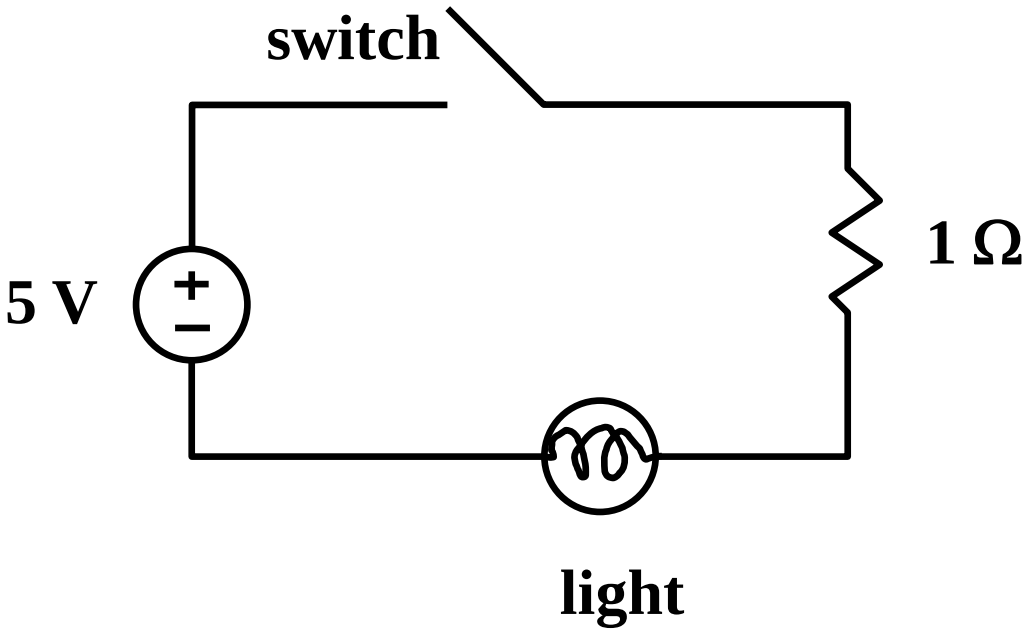
Diagram
Behaviour Diagram
Structure Diagram
Activity Diagram
State Machine Diagram
Use Case Diagram
Interaction Diagram
Communi-cation Diagram
Timing Diagram
Interaction Overview Diagram
Sequence Diagram
Class Diagram
Component Diagram
Object Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram


Use Case Diagrams
work flows
scenarios
structures
Activity Diagrams
Interaction Diagrams
inter-class behaviour
interaction sequences
between objects
Class Diagrams
structuring
intra-class
behavior
Package Diagrams
packaging
State Diagrams
Component Diagrams
nodal
Deployment Diagram

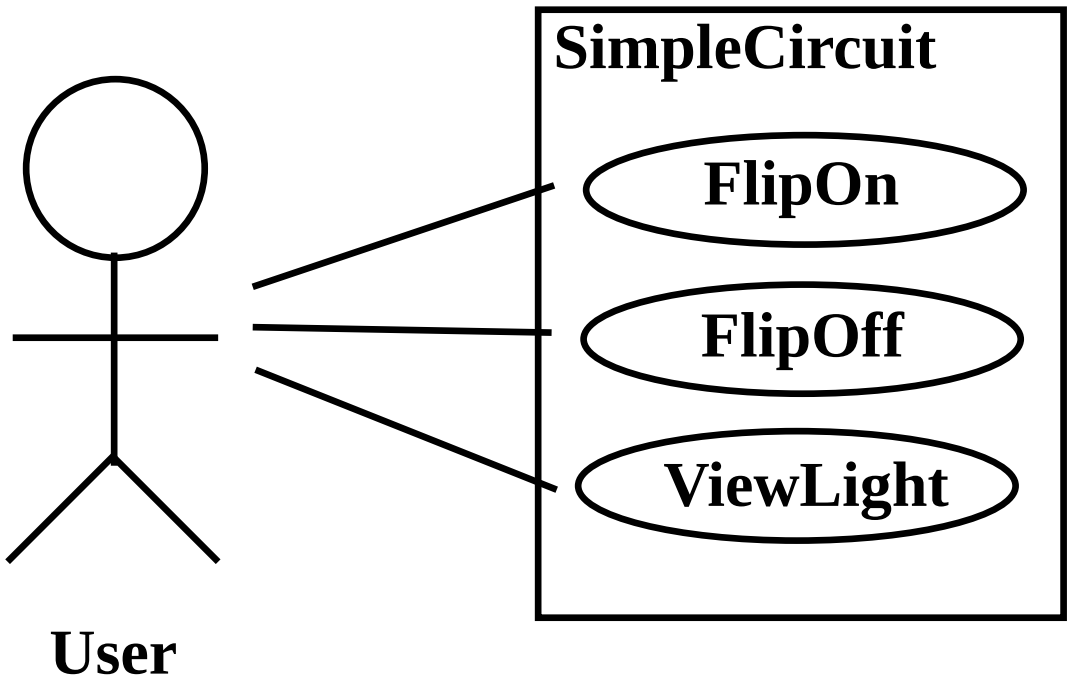
Three Software-Engineering Modeling Perspectives

- Analysis – for understanding
  - The objects represented in the models are real-world objects
  - Models focus on problem-domains concepts
  - They describe systems as they are
- Specification – for scoping and planning
  - The models include both real-world and software objects
  - The models show automation boundaries
  - The models describe what the system is to become
- Implementation – for designing / building
  - The objects in the models are mostly software objects
  - The models focus on solution-oriented concepts
  - The models describe what the software system is or will be

A Simple Problem

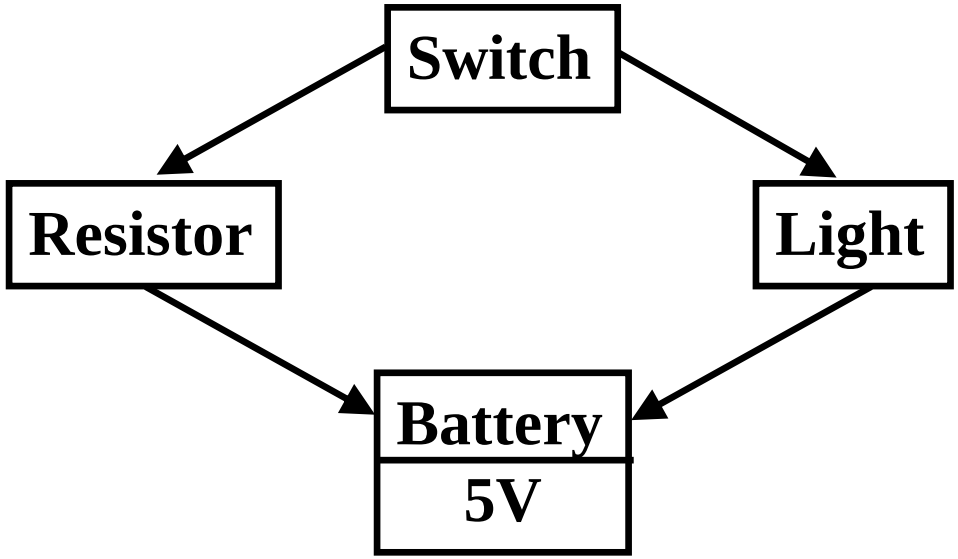


1. Use Case Diagram



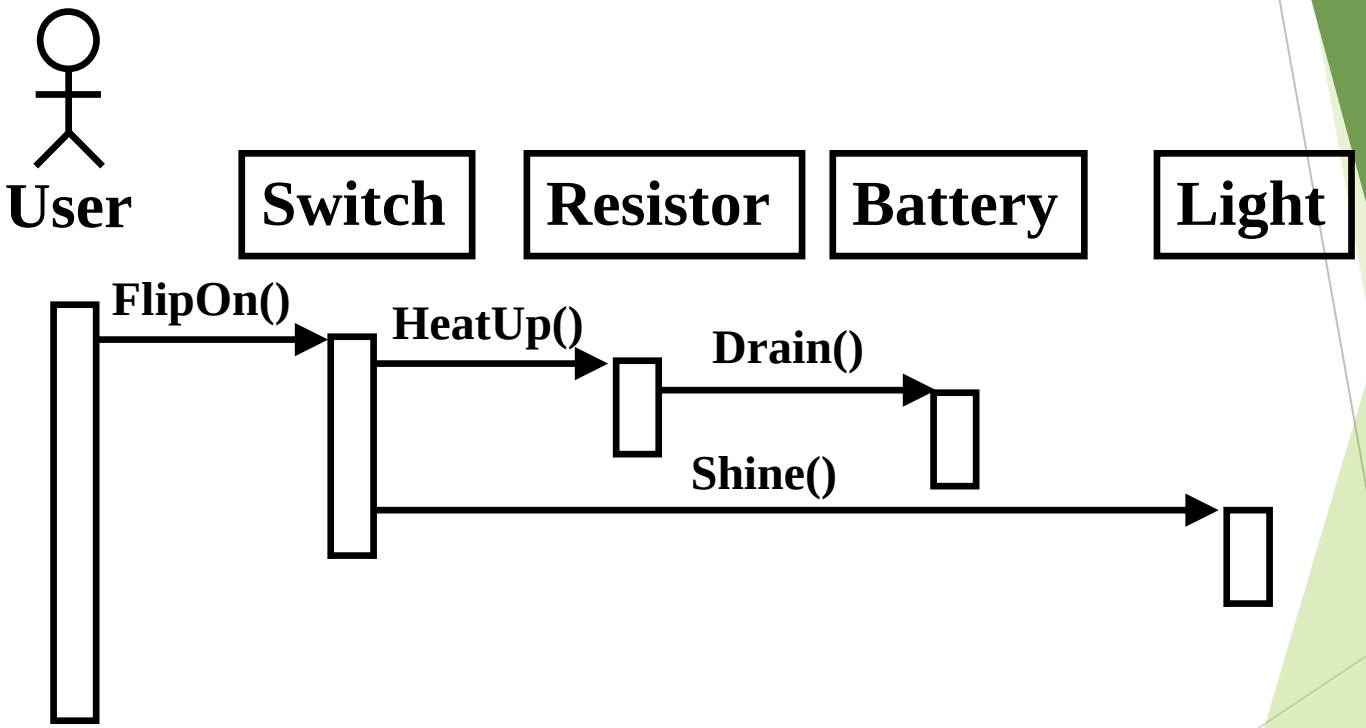
Functionality from user's point of view

2. Class Diagram



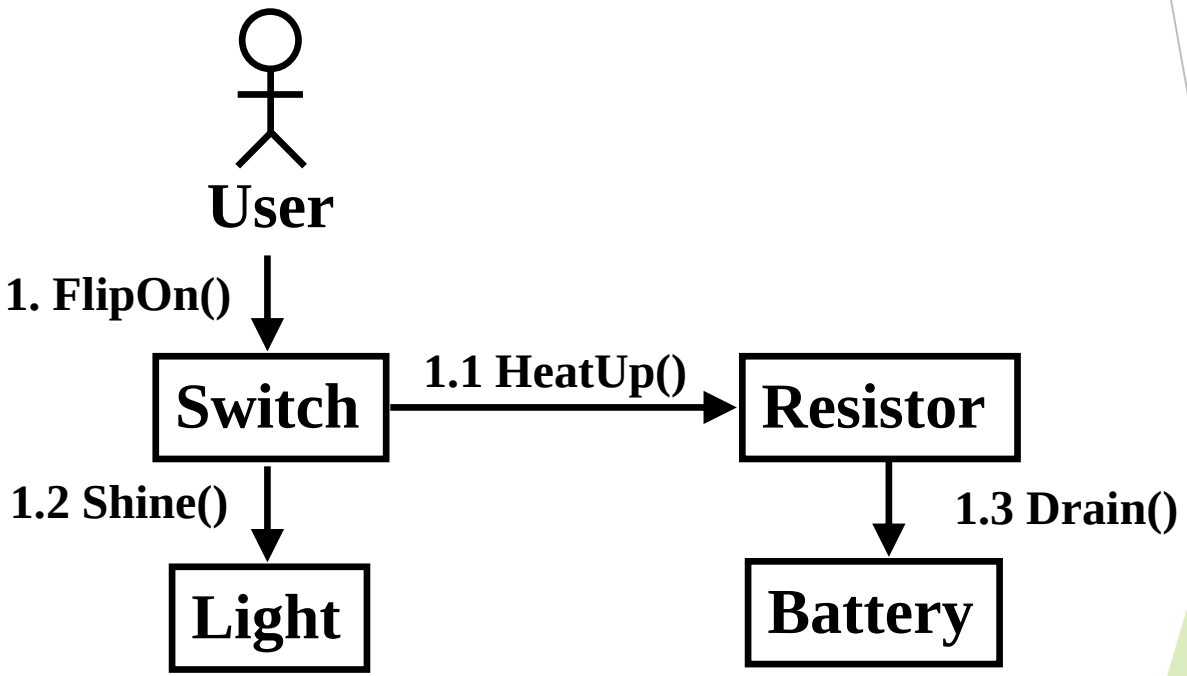
Structure of system (objects, attributes, associations, operations)

3. Interaction Diagram: (a) Sequence Diagram



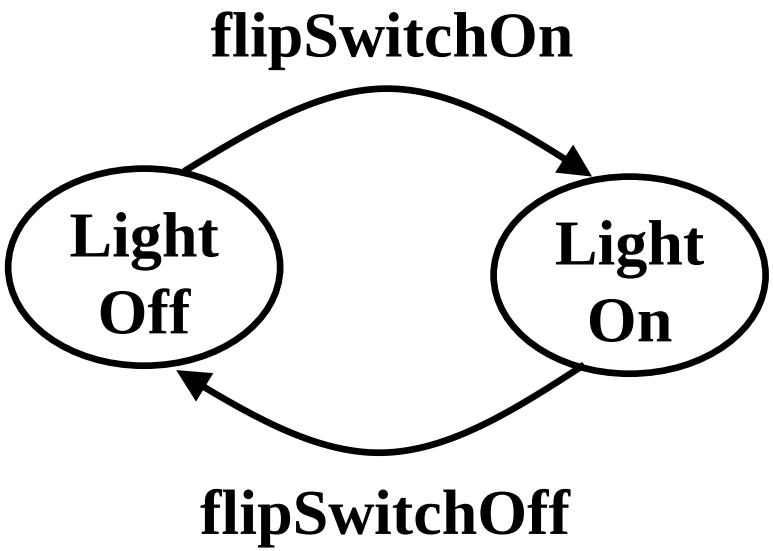
Messages between objects

3. Interaction Diagram: (b) Collaboration Diagram



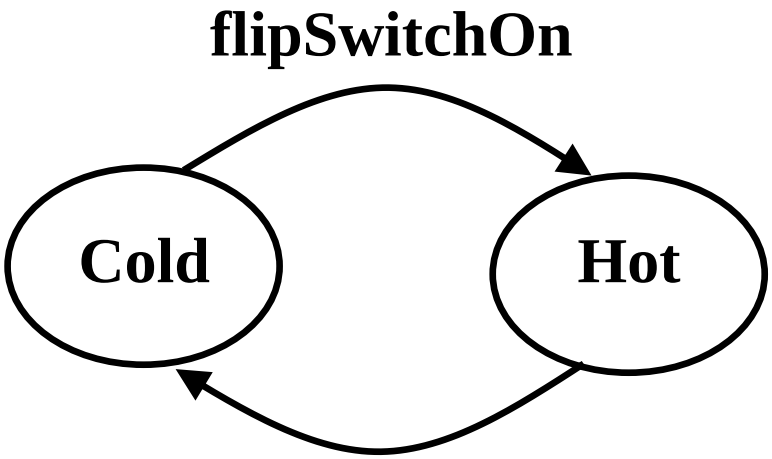
More compact, but harder to interpret

4. Statechart Diagram

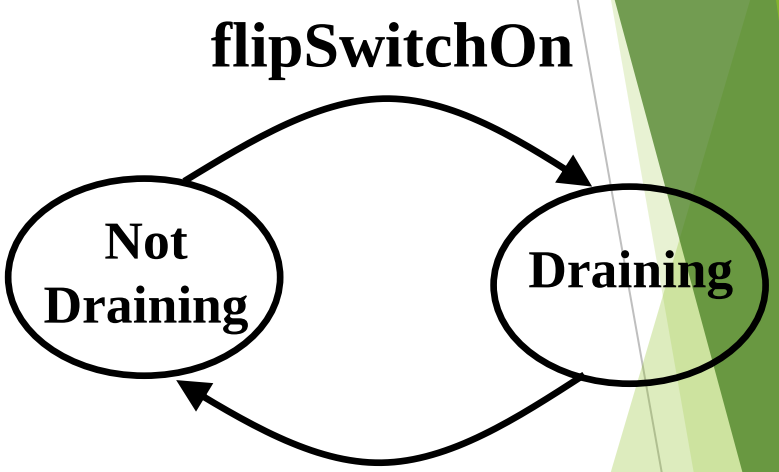


Transitions between states of one object
(Extension of Finite State Machine (FSM) model)

4. Statechart Diagram (different objects)

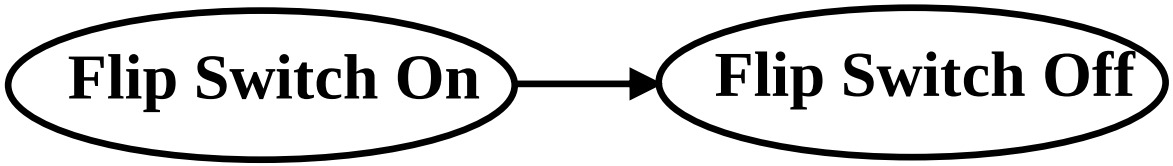


(Resistor)



(Battery)

5. Activity Diagram



Actions are states