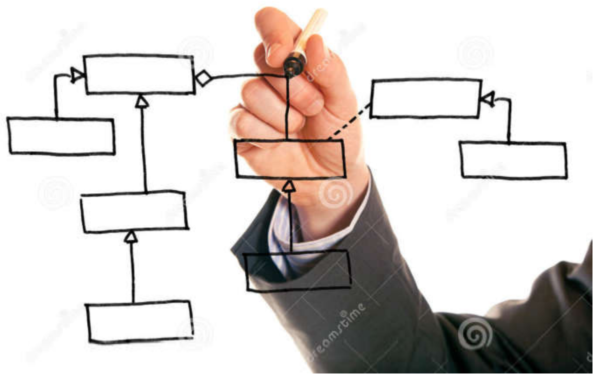
Notation Part 1

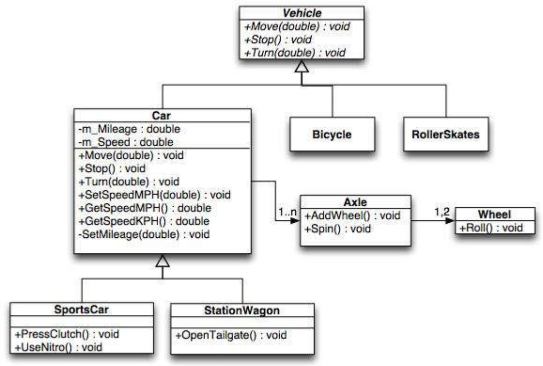
Object Orientated Analysis and Design

Benjamin Kenwright

Outline

- Review
- What do we mean by Notation?
- UML Diagrams
- Examples of Notation in Analysis and Design
- Summary/Discussion
- Conclusion





Revision Question

- Requirements analysis is critical to the success of a development project.

a) True

b) False

c) Depends upon the size of project

Answer

- Answer a)

Explanation: Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design

Question

- Requirements should specify ‘what’ but not ‘how’.

a) True

b) False

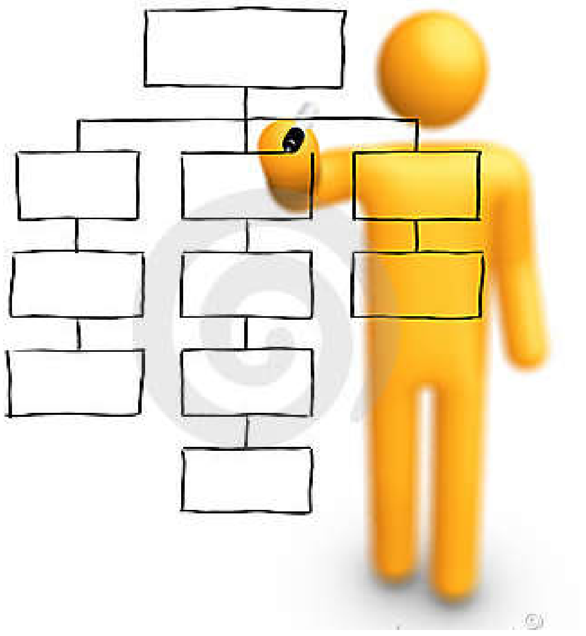
Answer

- Answer: a)

Explanation: ‘What’ refers to a system’s purpose, while ‘How’ refers to a system’s structure and behavior.

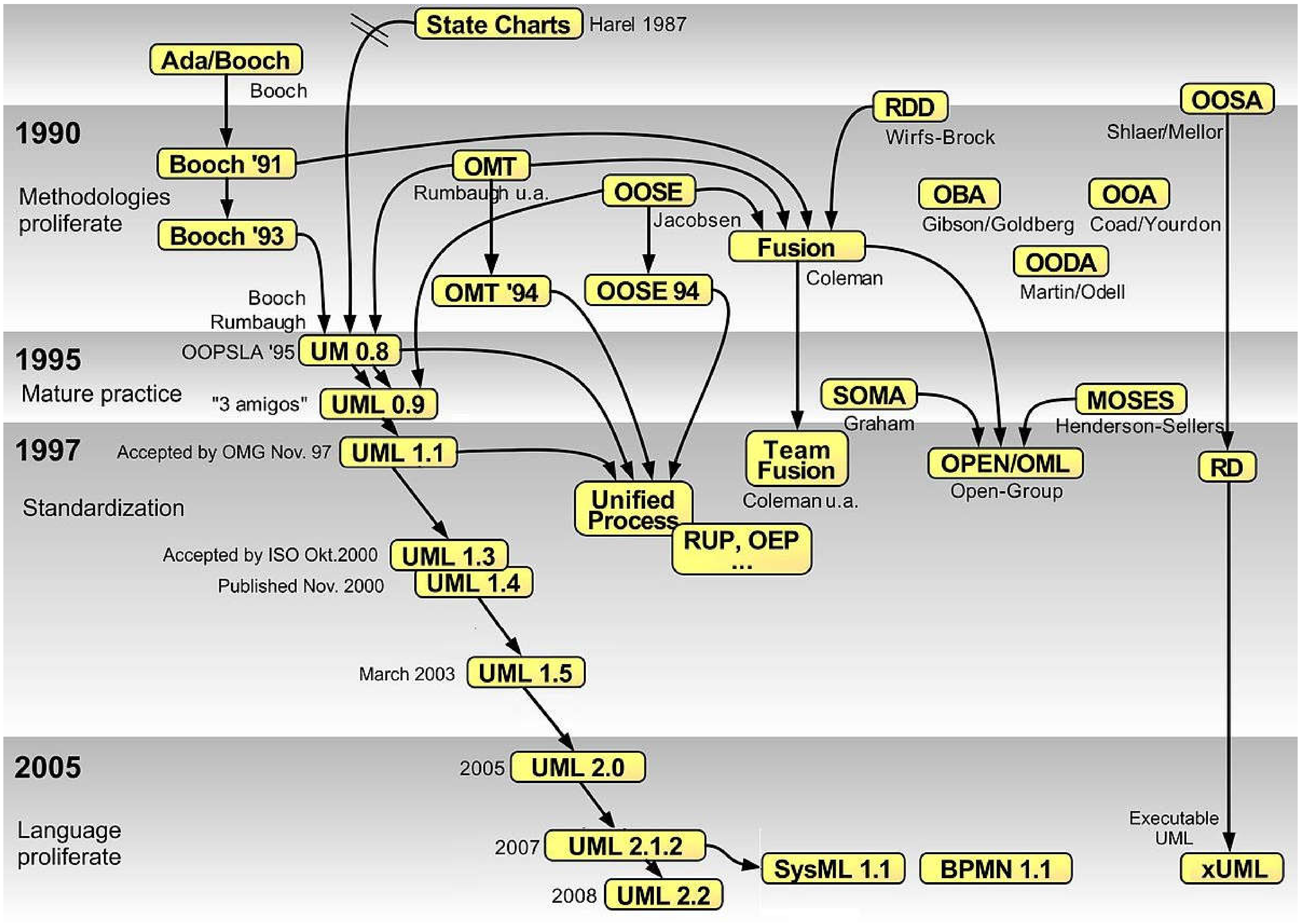
Diagrams

- Drawing a diagram does not constitute analysis or design
- However, diagrams provide visual aids
  - ▷Clarifying the concept
  - ▷Various forms, e.g., 3d-software, whiteboards, napkins, and the backs of envelopes

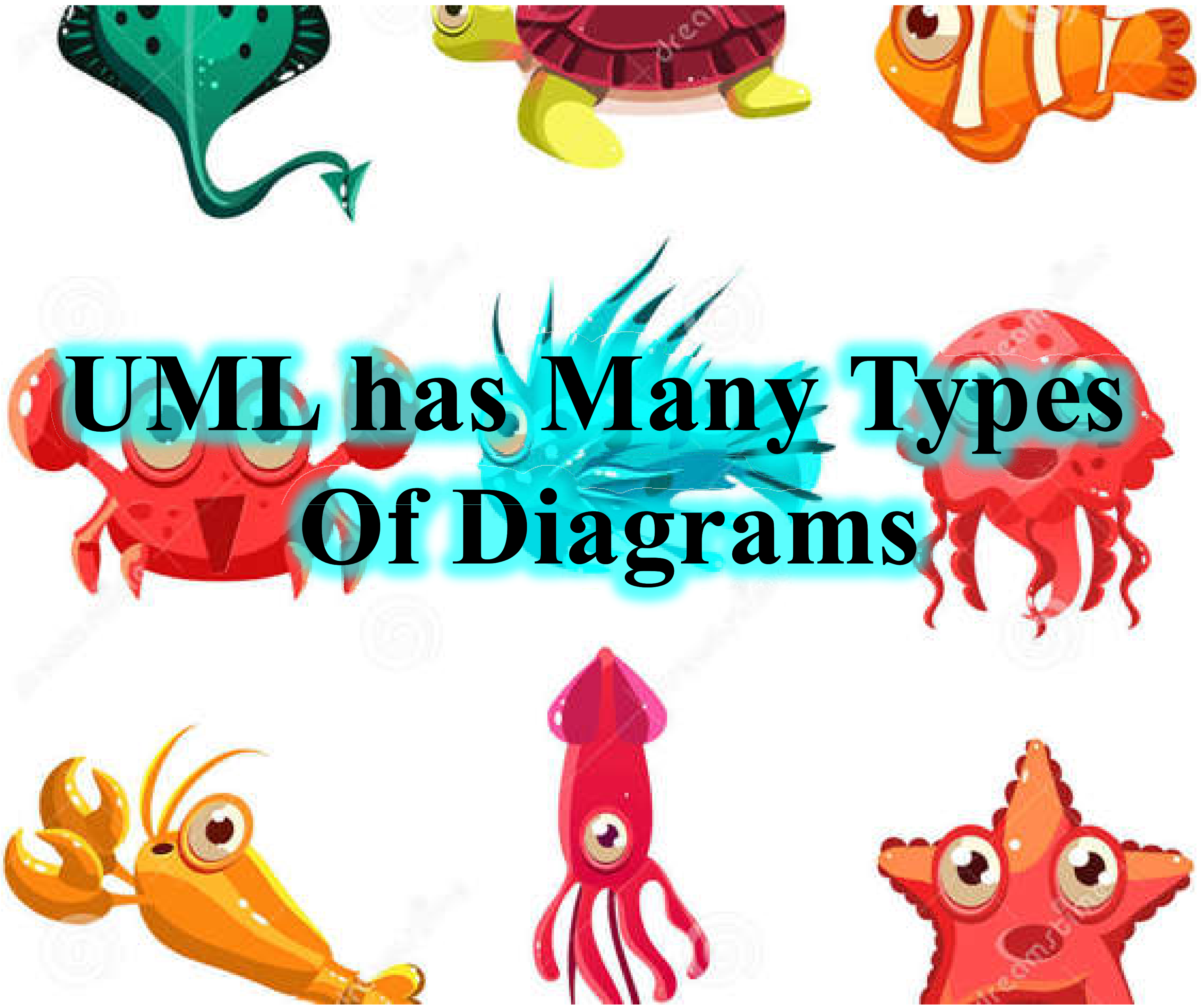


The Unified Modeling Language (UML)

- **Primary** modeling language used to analyze, specify and design software systems
- UML is used to model (i.e., represent) the system being built



State Charts
Harel 1987
Ada/Booch
Booch
1990
Booch '91
Methodologies proliferate
Booch '93
OMT
Rumbaugh u.a.
OOSE
Jacobsen
RDD
Wirfs-Brock
OOSA
Shlaer/Mellor
OBA
Gibson/Goldberg
OOA
Coad/Yourdon
Fusion
Coleman
OODA
Martin/Odell
OMT '94
OOSE 94
Booch
Rumbaugh
1995
OOPSLA '95
UM 0.8
Mature practice
"3 amigos"
UML 0.9
SOMA
Graham
MOSES
Henderson-Sellers
1997
Accepted by OMG Nov. 97
UML 1.1
Unified Process
Team Fusion
Coleman u.a.
OPEN/OML
Open-Group
RD
Standardization
RUP, OEP ...
Accepted by ISO Okt.2000
UML 1.3
Published Nov. 2000
UML 1.4
March 2003
UML 1.5
2005
2005
UML 2.0
Language proliferate
2007
UML 2.1.2
SysML 1.1
BPMN 1.1
Executable UML
xUML
2008
UML 2.2

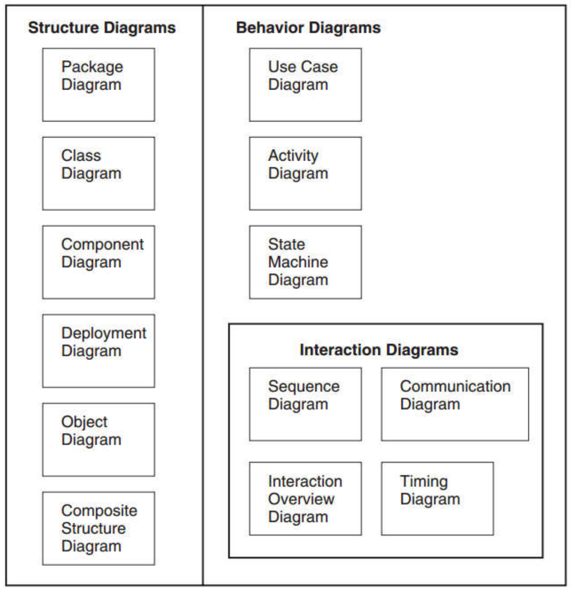


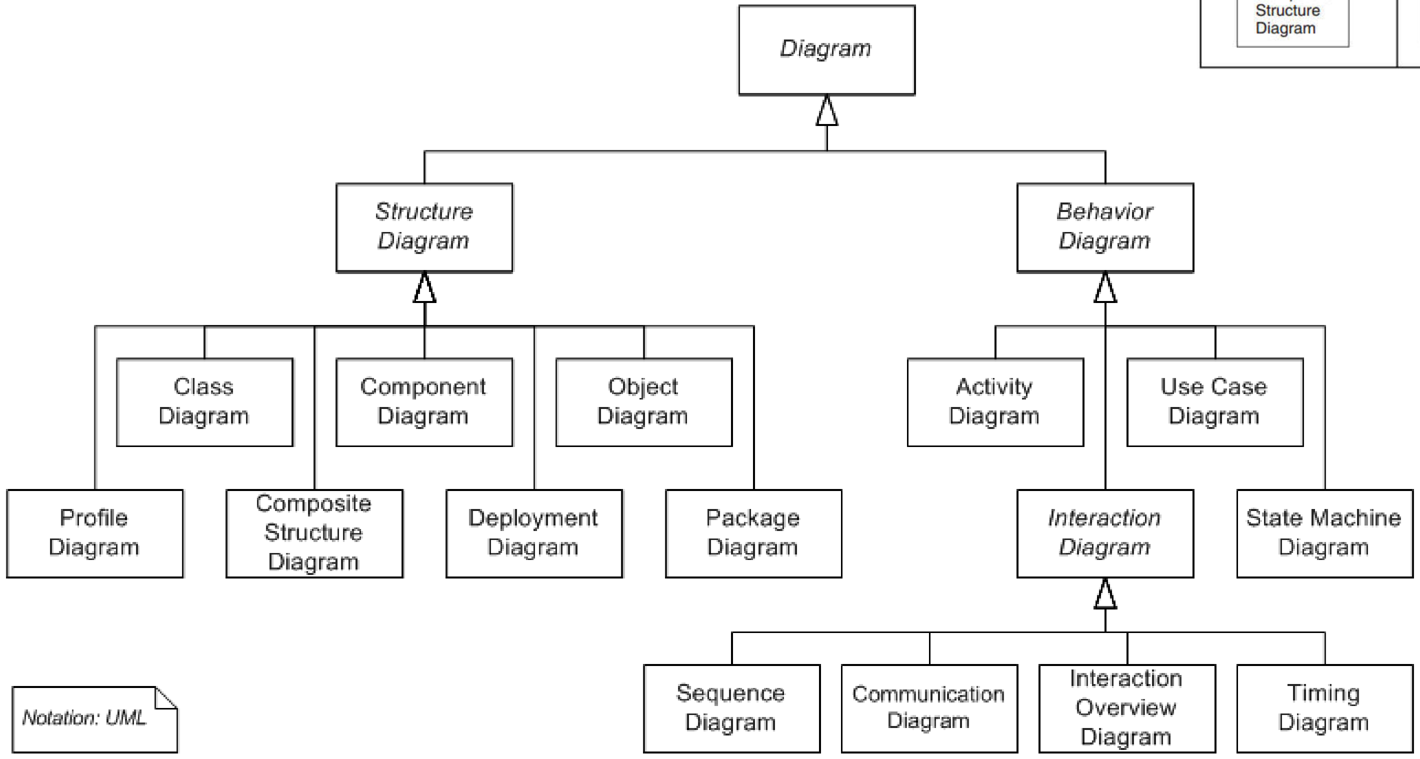
UML has Many Types Of Diagrams

UML

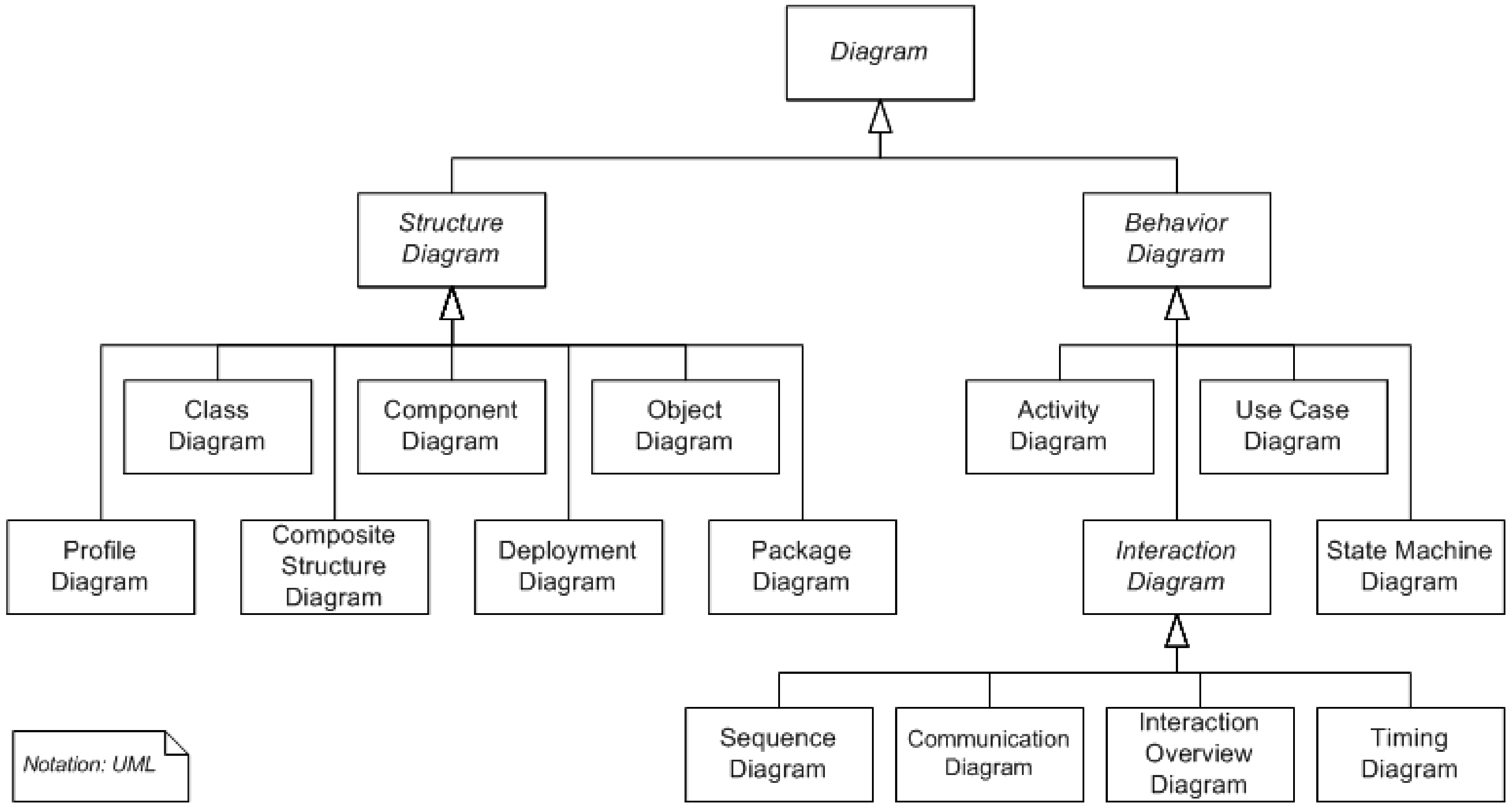
- Unified Modeling Language (UML): is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed and was created by the Object Management Group (OMG)
- UML includes a set of **graphic notation** techniques to create visual models of object-oriented software-intensive systems.
- UML defines different types of diagrams: class (package), object, use case, sequence, collaboration, activity, component, and deployment.
- UML has various types of diagrams divided into **two main categories**

UML Diagrams





UML Diagrams



Two UML Views of System Model

- **Static (or structural)** view: emphasizes the static structure of the system using objects, attributes, operations and relationships. It includes class diagrams and composite structure diagrams.
- **Dynamic (or behavioral)** view: emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams

Two UML Views of System Model

Structure Diagrams

- Package Diagram
- Component Diagram
- Deployment Diagram
- Composite Structure Diagrams
- Class Diagram
- Sequence Diagram
- Object Diagrams

Behavior Diagrams

- Use Case Diagram
- Activity Diagram
- State Machine Diagrams

Interaction Diagrams

- Communication Diagrams
- Interaction Overview Diagrams
- Timing Diagrams

Question

- Is a Class Diagram a static or dynamic system model view?

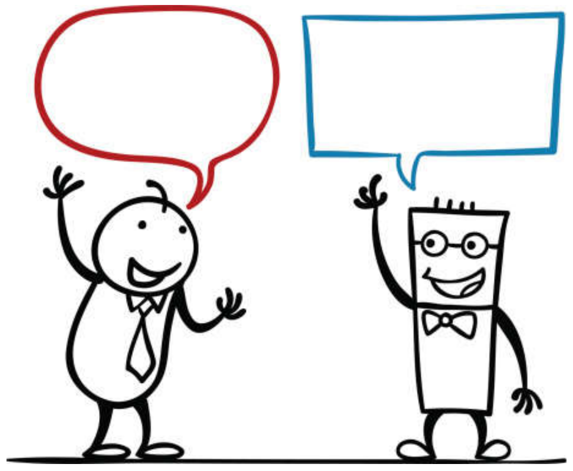
a) Static (or Structural)
b) Dynamic (or Behavioral)

Answer

a) Static (or Structural)

Activity

- Write down as many UML diagram views as you can remember
- Also state if they're Static (or Structural) or Dynamic (or Behavioral)



Structure Diagrams

- Structure diagrams emphasize the things that must be present in the system being modelled
- Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.
- For example, the component diagram describes how a software system is split up into components and shows the dependencies among these components.

Behavior Diagrams

- Behavior diagrams emphasize what must **happen** in the system being modelled
- Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems
- As an example, the activity diagram describes the business and operational step-by-step activities of the components in a system

Plan

- Review diagrams in an order in which one might typically develop them
  - i.e., instead of reviewing all the structural and then behavioral diagrams seperately

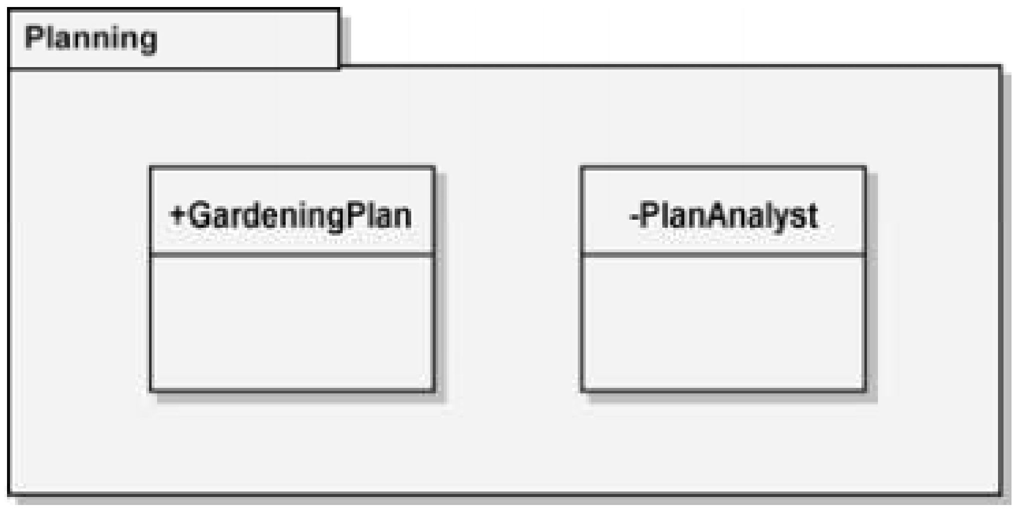
Package Diagrams

- **Package Diagrams**
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Visibility

- Public (+) Visible to elements within its containing package, including nested packages, and to external elements
- Private (-) Visible only to elements within its containing package and to nested packages

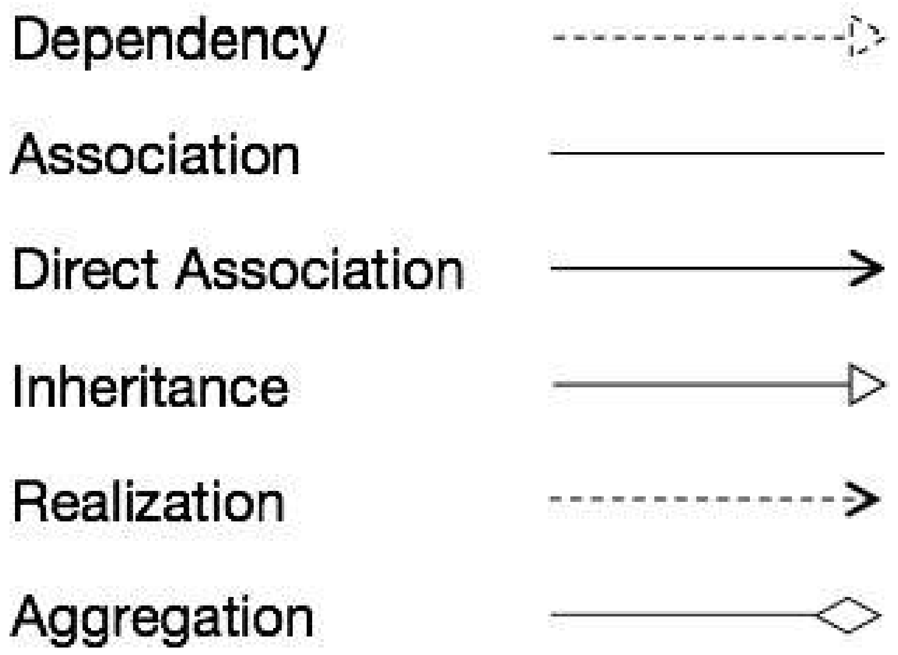
Example



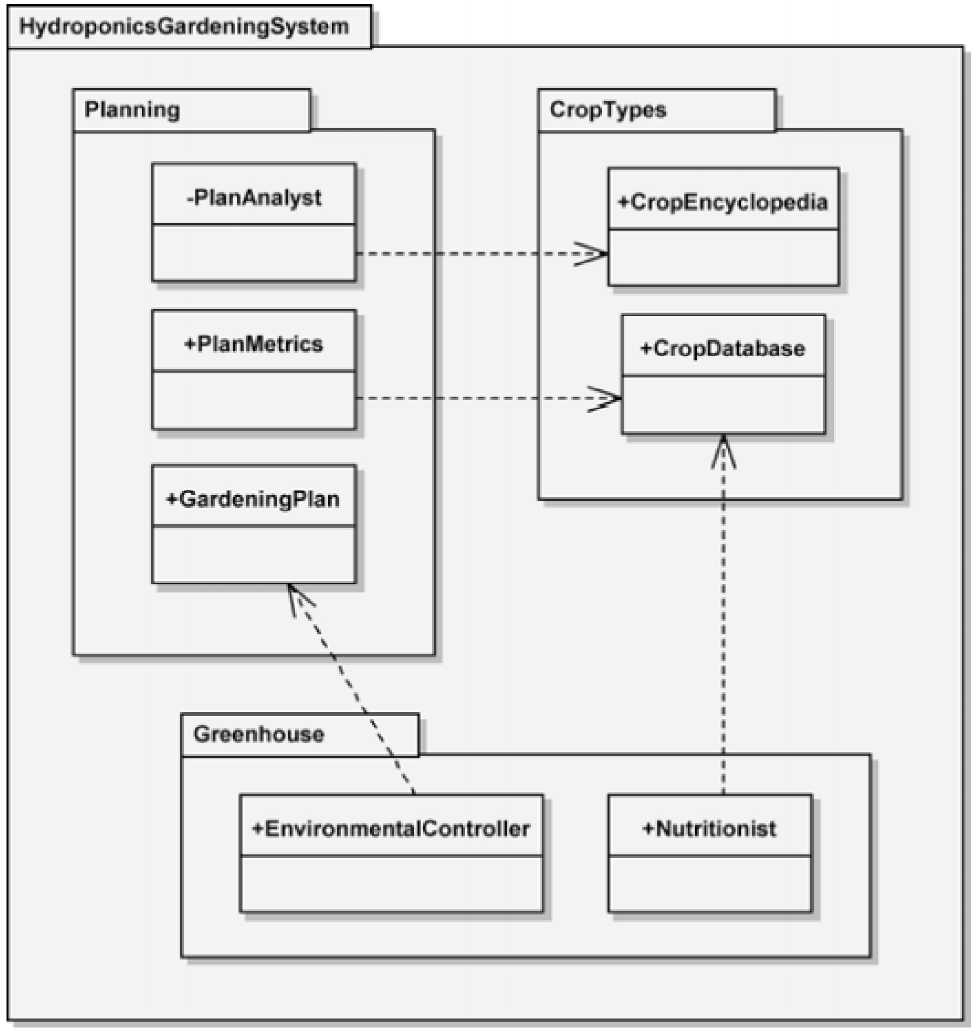
The Visibility of Elements within `Planning` Package

Dependencies between UML elements (including Packages)

- Notations for the different types of relationships are as follows



Example


HydroponicsGardeningSystem
Planning
-PlanAnalyst
+PlanMetrics
+GardeningPlan
CropTypes
+CropEncyclopedia
+CropDatabase
Greenhouse
+EnvironmentalController
+Nutritionist

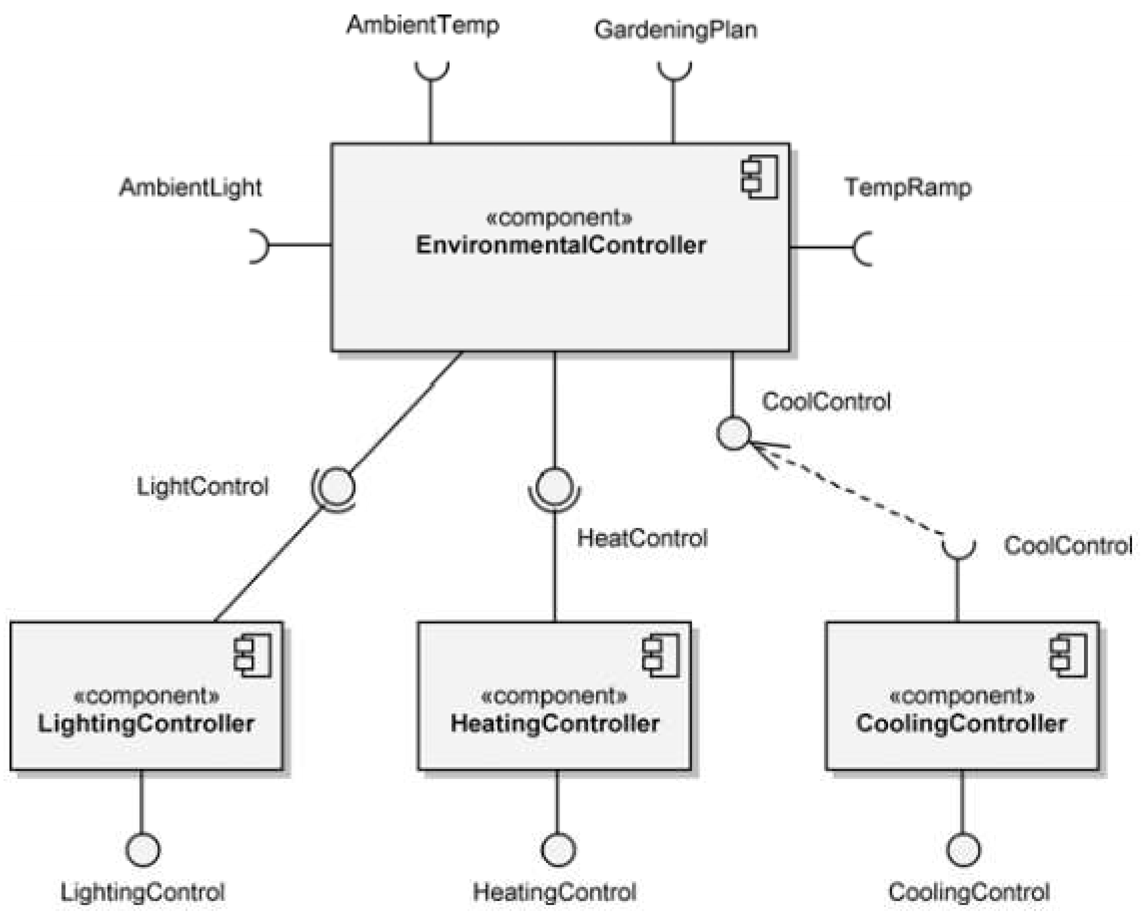
Component Diagrams

- Package Diagrams
- **Component Diagrams**
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Component Notation

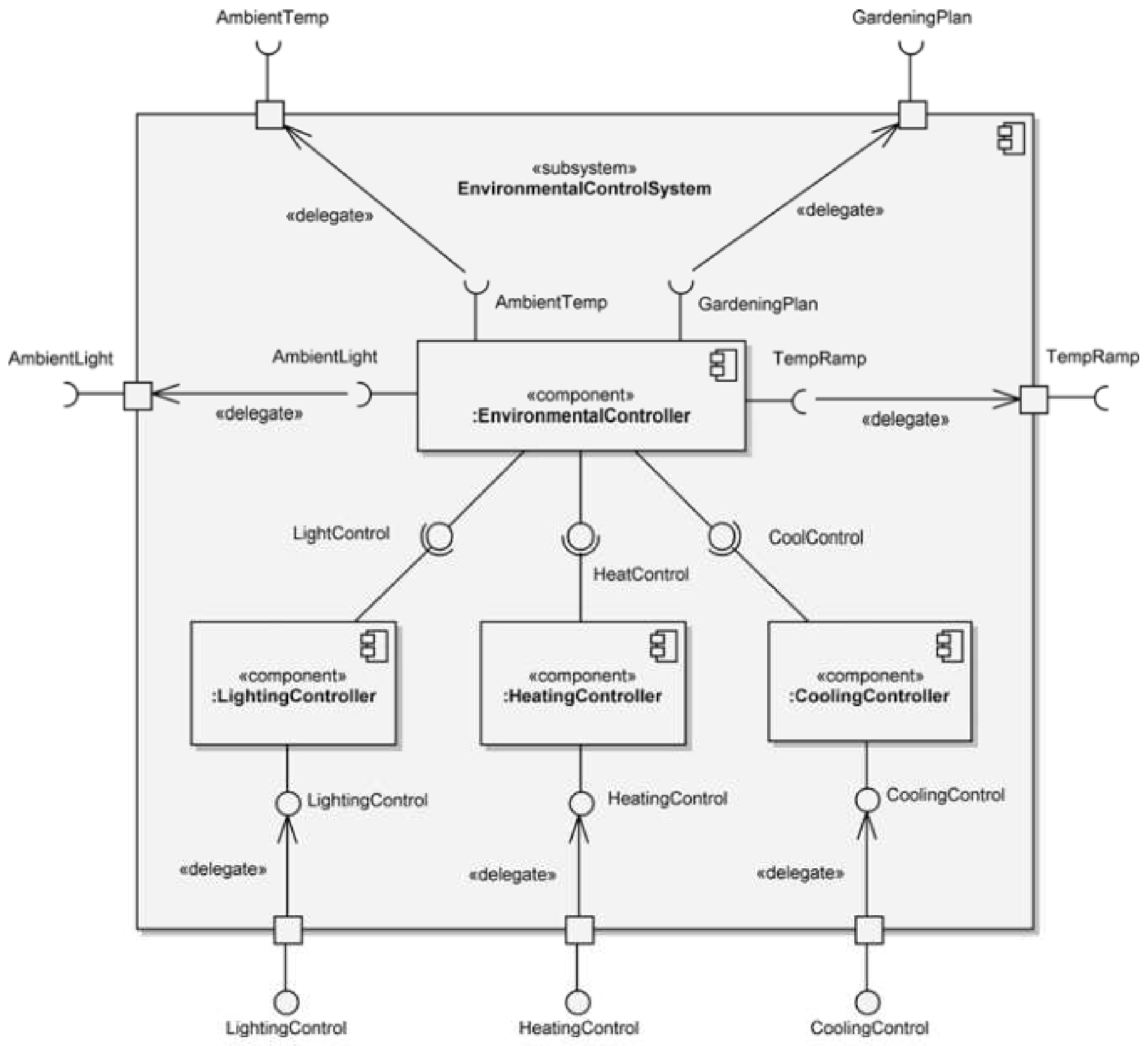
- Component is a structured classifier, its detailed assembly can be shown with a composite structure using
  - Parts
  - Ports, and
  - Connectors

Example



The Component Diagram for `EnvironmentalControlSystem`

A Component's Internal Structure



The Internal Structure of `EnvironmentalControlSystem`

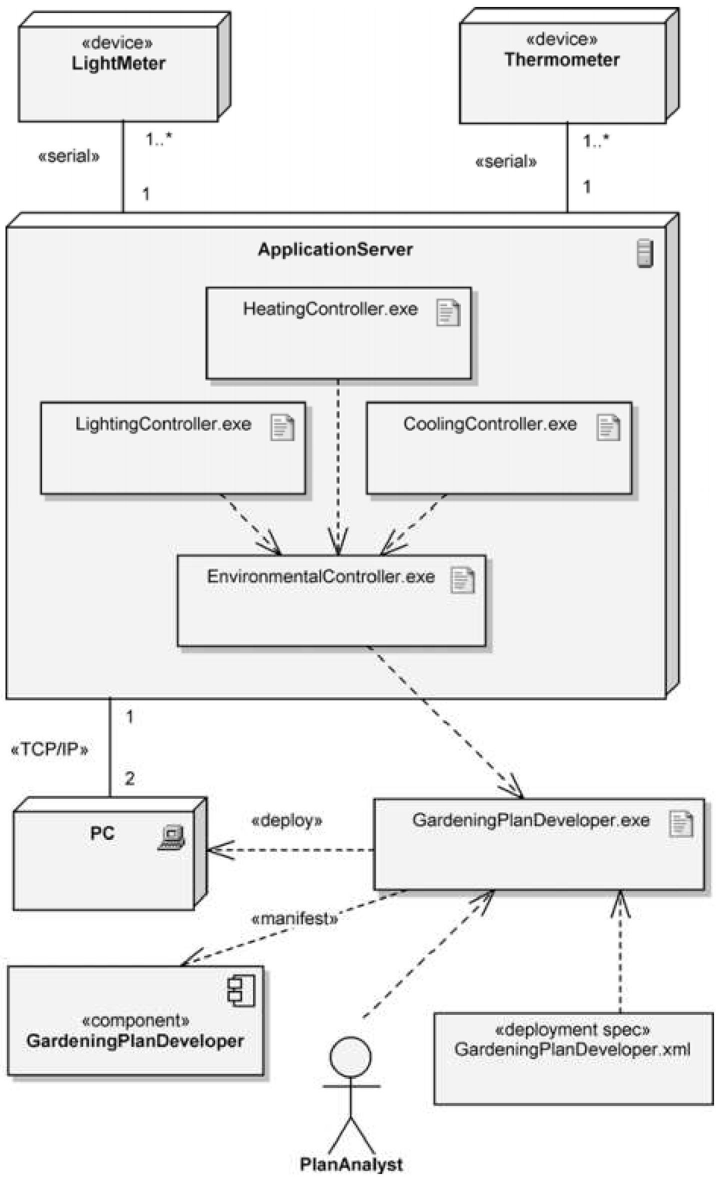
Deployment Diagrams

- Package Diagrams
- Component Diagrams
- **Deployment Diagrams**
- Use Case Diagrams
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Deployment Diagram

- A deployment diagram is used to show the allocation of artifacts to nodes in the physical design of a system.
- A single deployment diagram represents a view into the artifact structure of a system.
- The three essential elements of a deployment diagram are artifacts, nodes, and their connections.

Example



The Deployment Diagram for `EnvironmentalControlSystem`

Question

- The three essential elements of a deployment diagram are:

a) nodes, connections and their elements

b) artifacts, nodes, and their connections

c) elements, relationships and connectors

d) inheritance, relationships and connectors

Answer

b) artifacts, nodes, and their connections

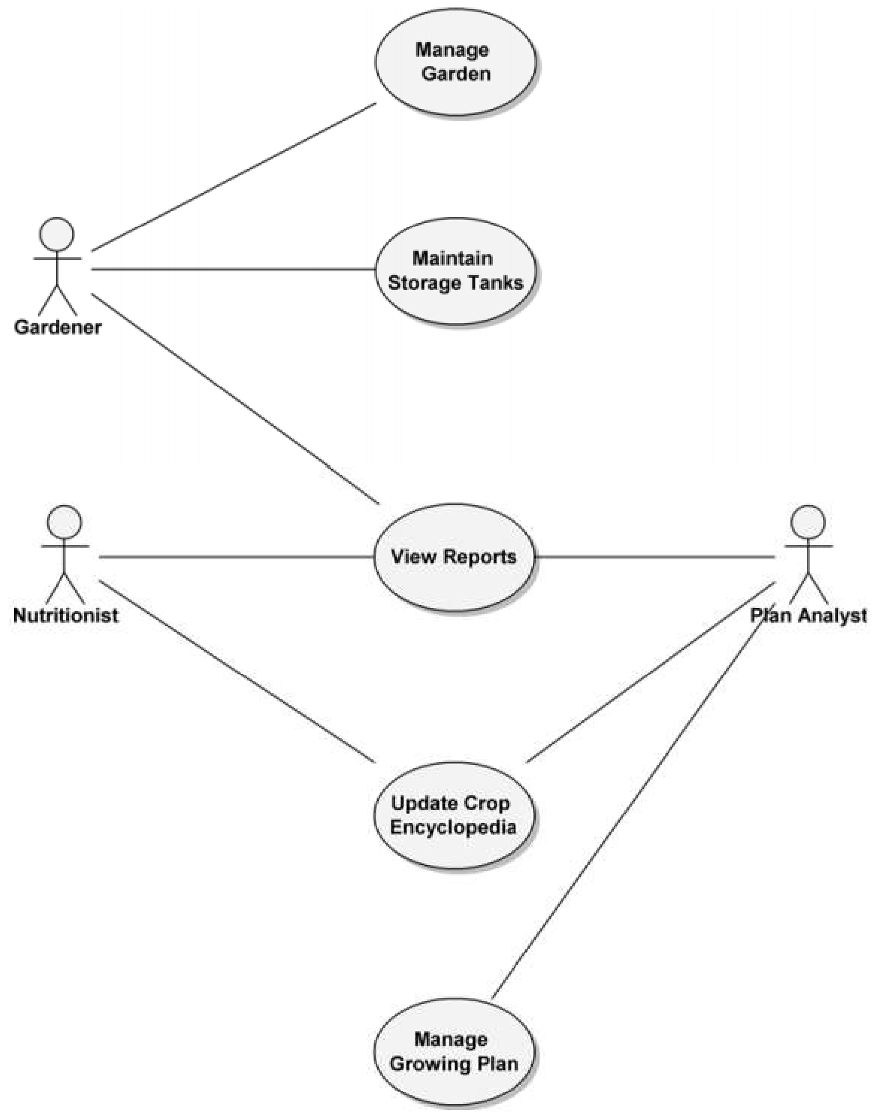
Use Case Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- **Use Case Diagrams**
- Activity Diagrams
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Use Case Diagrams

- Use case diagrams are used to depict the context of the system to be built and the functionality provided by that system.
- They depict who (or what) interacts with the system.
- They show what the outside world wants the system to do.

Example



A Use Case Diagram

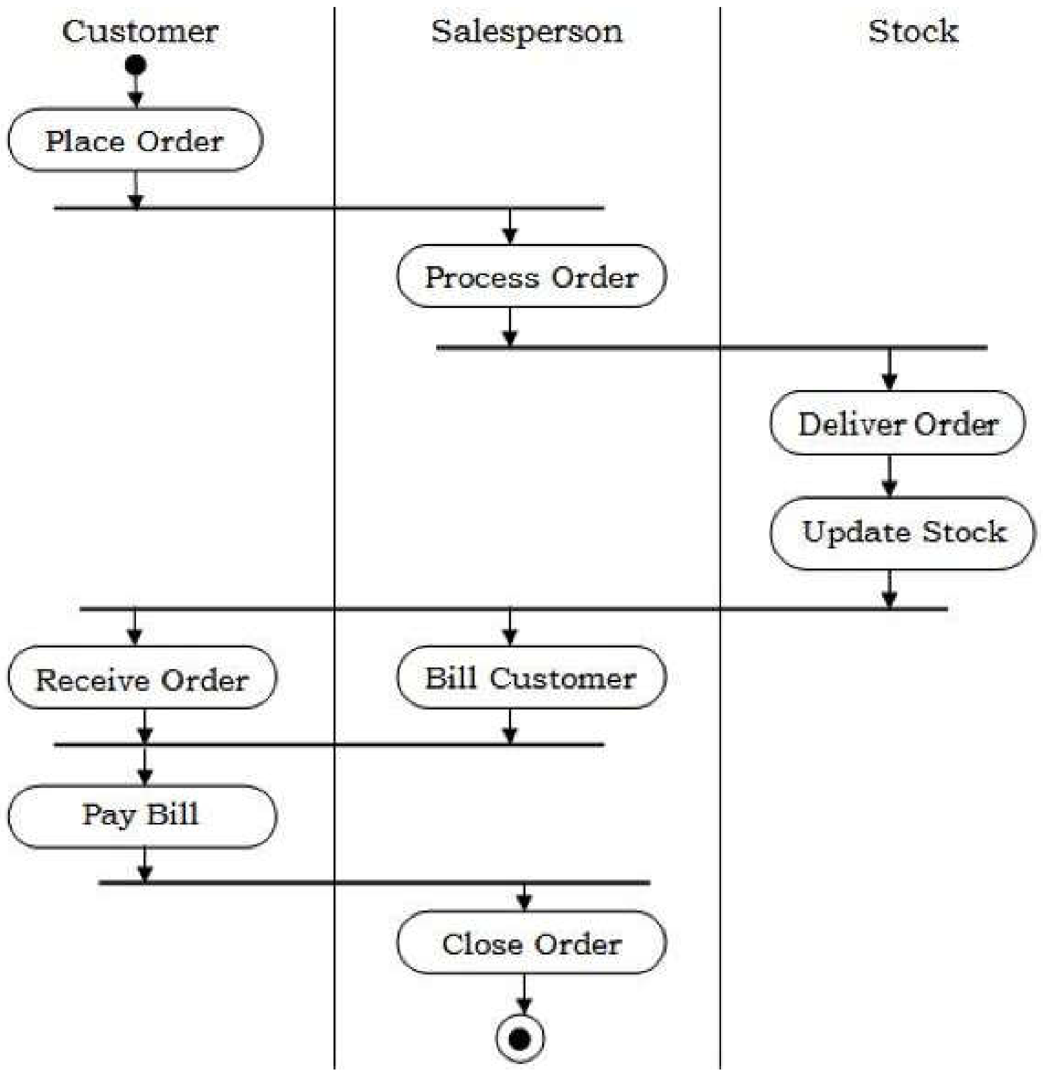
Activity Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- **Activity Diagrams**
- Class Diagrams
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Activity Diagram

- Activity diagrams provide visual depictions of the flow of activities, whether in a system, business, workflow, or other process
- These diagrams focus on the activities that are performed and who (or what) is responsible for the performance of those activities.

Example


Customer
Salesperson
Stock
Place Order
Process Order
Deliver Order
Update Stock
Receive Order
Bill Customer
Pay Bill
Close Order

Class Diagrams

- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- **Class Diagrams**
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

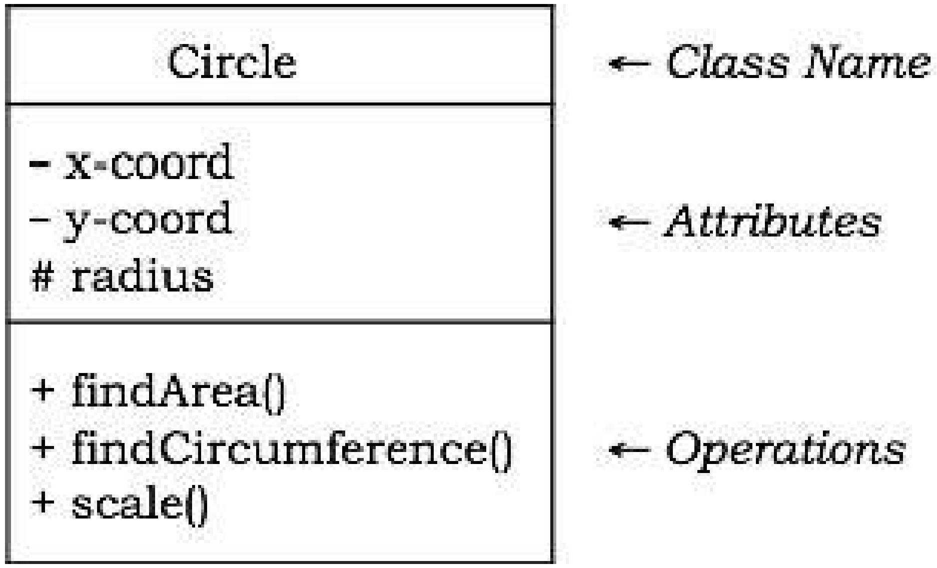
Class Diagram

- A class diagram is used to show the existence of classes and their relationships in the logical view of a system

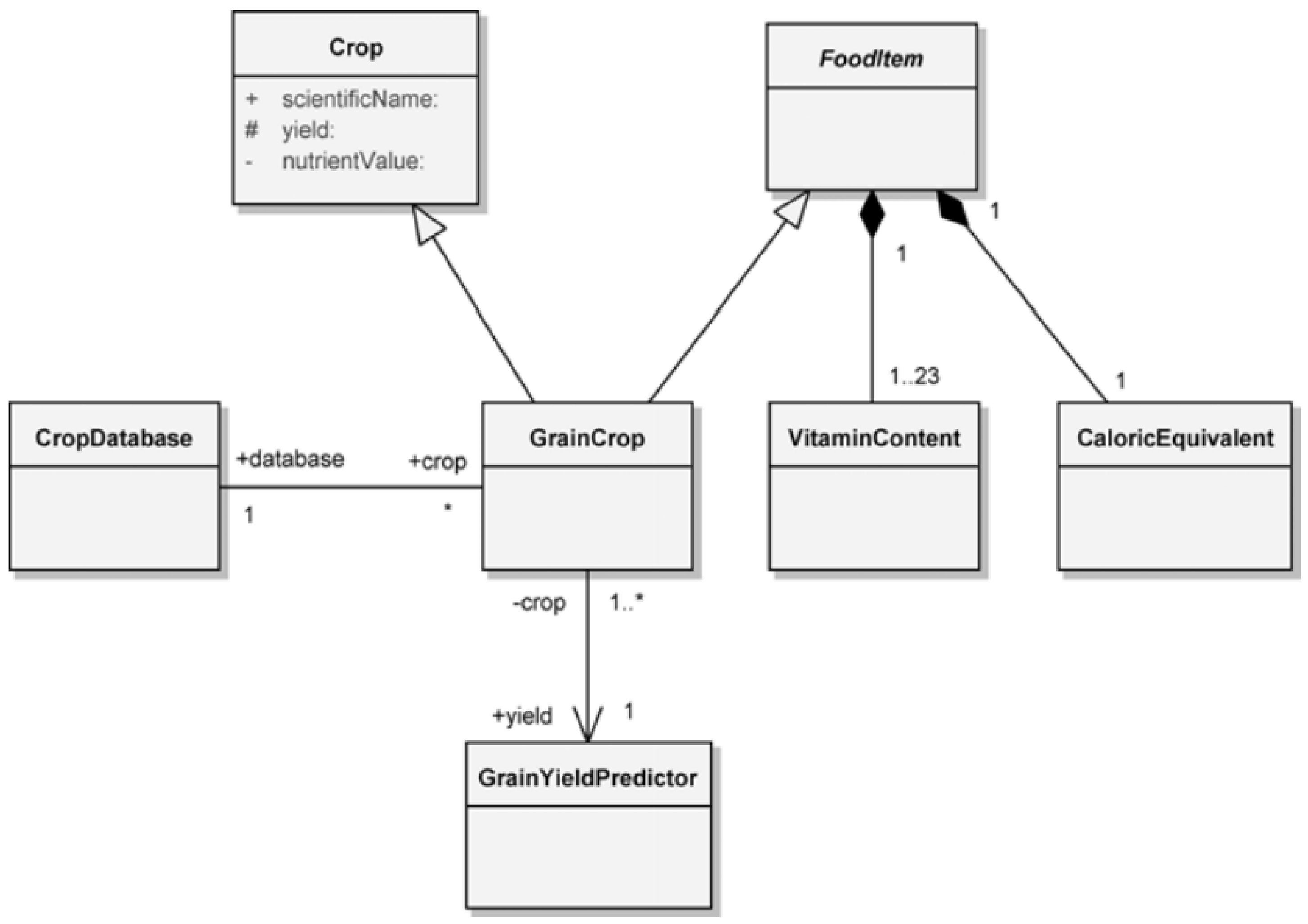
Visibility

- Public (+) Visible to any element that can see the class
- Protected (#) Visible to other elements within the class and to subclasses
- Private (-) Visible to other elements within the class
- Package (~) Visible to elements within the same package

Diagrammatic Representation of a Class



Example



Crop
+ scientificName:
# yield:
- nutrientValue:
FoodItem
1
1
1..23
1
CropDatabase
+database
1
+crop
*
GrainCrop
VitaminContent
CaloricEquivalent
-crop
1..*
+yield
1
GrainYieldPredictor

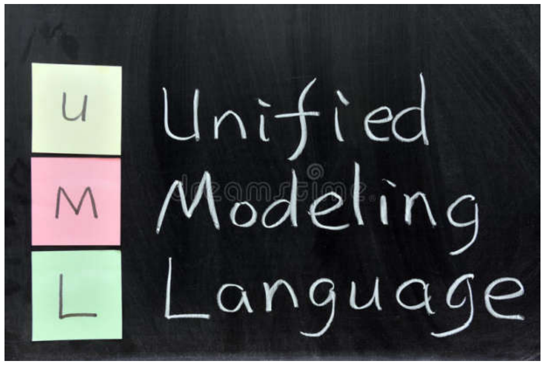
UML Diagrams

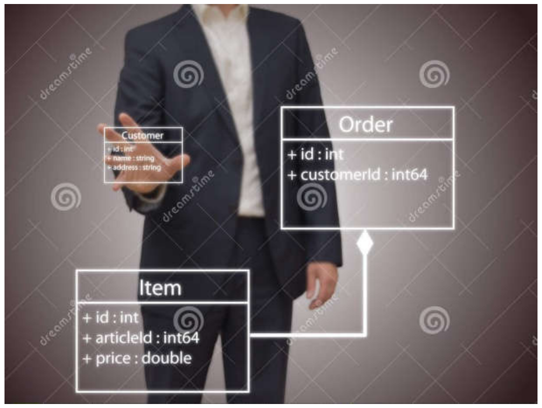
- Package Diagrams
- Component Diagrams
- Deployment Diagrams
- Use Case Diagrams
- Activity Diagrams
- **Class Diagrams**
- Sequence Diagrams
- Interaction Overview Diagrams
- Composite Structure Diagrams
- State Machine Diagrams
- Timing Diagrams
- Object Diagrams
- Communication Diagrams

Next Week

Summary

- Fundamentals of Notation in Object Orientated Analysis and Design
  - ▷Specifically UML and the Different Models/Diagrams
- Designing is not the act of drawing a diagram; a diagram simply captures a design





This Week

- Review Slides
- Read Chapters 5 and 6
- Online Quizzes
- Version Control (GitHub)

Questions/Discussion