Unified Modeling Language

LESSON 13

Introduction

Outline

1. What’s in a Modeling Language
2. Models and Diagrams
3. Degrees of UML
4. Views of Your Model
5. Dispelling Misconceptions about UML

Overview

In this chapter, you are going to learn about

- Definition of Unified Modeling Language (UML)
- Roles of Models and Diagrams
- Know different types of Views of Models
- Clearing the wrong understanding of the use of UML

Learning content

1. What’s in a Modeling Language
    1. What is UML?
    2. 6 main advantages of UML
    3. Verbosity, Ambiguity, Confusion: Modeling with Informal Languages
    4. Detailed Overload: Modeling with Code
    5. Getting the Balance Right: Formal Languages
2. Models and Diagrams
    1. Model in big picture
    2. Roles of Diagram
    3. Relation of Models and Diagrams
3. Degrees of UML
    1. UML as a Sketch
    2. UML as a Blueprint
    3. UML as a Programming Language
4. Views of Your Model
    1. Use case view
    2. Process view
    3. Logical view
    4. Development view
    5. Physical view
5. Dispelling Misconceptions about UML
    1. When to draw diagrams
    2. When NOT to draw diagrams
    3. Example of a medium project

Learning objectives

Upon completion of this chapter, you will be able to

- Define Unified Modeling Language (UML)
- Identify the roles of Models and Diagrams
- Define the use of UML in software
- List the views of your model
- Omit wrong use of UML

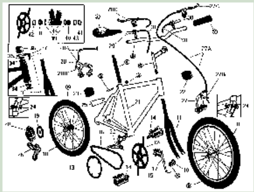
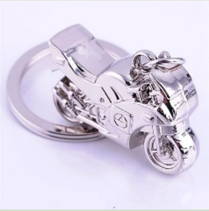
Keywords

Keywords	Description
UML	is short for Unified Modeling Language, the standard modeling language for software and systems development
Unified	only one, standard, uniform.
Modeling	Use as an example to follow or imitate.
Language	(terminology, syntax and convention that we have to respect).
Diagram	a simplified drawing to demonstrate (ex pie chart, bar chart)

Pre-Test

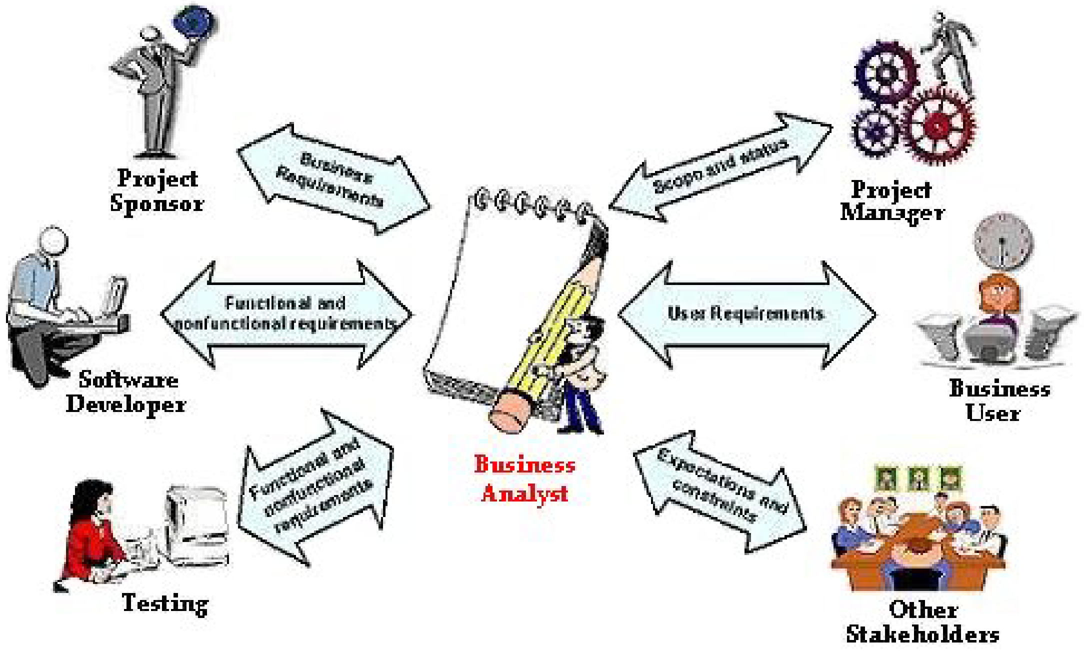
Question	Possible answers	Correct Answer	Feedback of the question
What is **computer program**?	1. A plan of actions to reach an objective 2. A series of instructions given to a computer to direct it to carry out certain operations	2. A series of instructions given to a computer to direct it to carry out certain operations	
What is **programming language**?	1. Language used to create program (s) 2. Language that speak to a program	1	

Pre-Test

Question	Possible answers	Correct Answer	Feedback of the question
Which one is Model of Bicycle?	1.  2. 	1	

1.1. What is UML?

- Unified Modeling Language (UML) is the standard modeling language for software and systems development. It consists of an integrated set of diagrams developed to help accomplish the following tasks:
  - Specification
  - Visualization
  - Architecture design
  - Construction
  - Simulation and Testing
  - Documentation



1.2. Main advantages of UML

- It’s a formal language
- It’s concise
- It’s comprehensive
- It’s scalable
- It’s built on lessons learned
- It’s the standard

Every IT school includes UML in their programs.

Short and clear

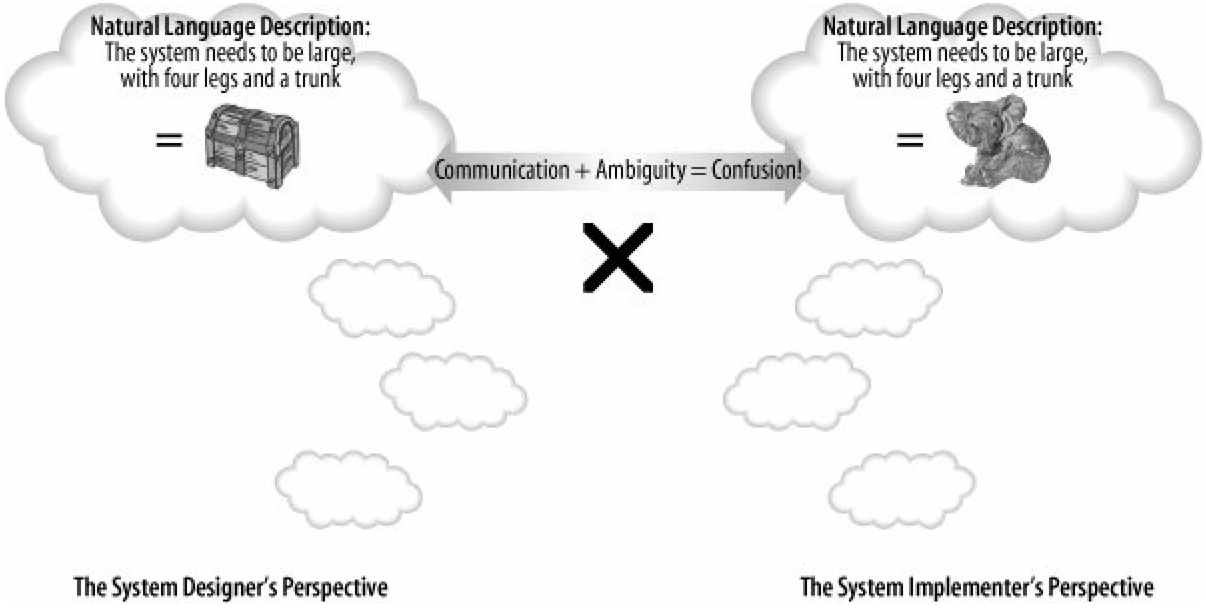
Image+text is easier to understand than plain text.

We can add more diagrams and more information

Built by experienced large companies

Standardized by Object Management Group (OMG).
In 2005, it is published as ISO Standard.

1.3. Verbosity, Ambiguity, Confusion: Informal Languages

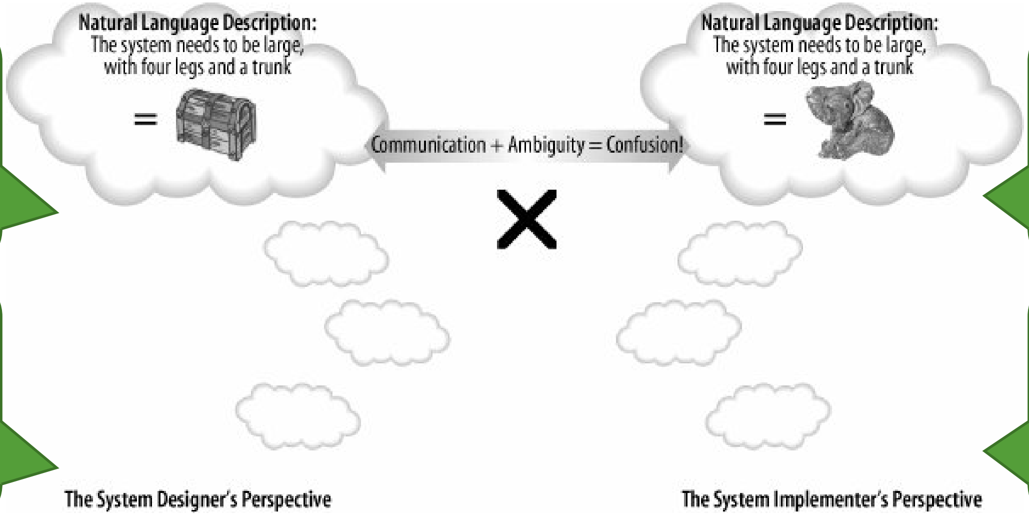

Natural Language Description:
The system needs to be large, with four legs and a trunk
=
Communication + Ambiguity = Confusion!
Natural Language Description:
The system needs to be large, with four legs and a trunk
=
The System Designer's Perspective
The System Implementer's Perspective

1.3. Verbosity, Ambiguity, Confusion: Informal Languages

"**The system needs to be large,**" a big box is **large**.

"**with four legs**" the box has 4 legs.

"**and a trunk**" the box has body for storing things.

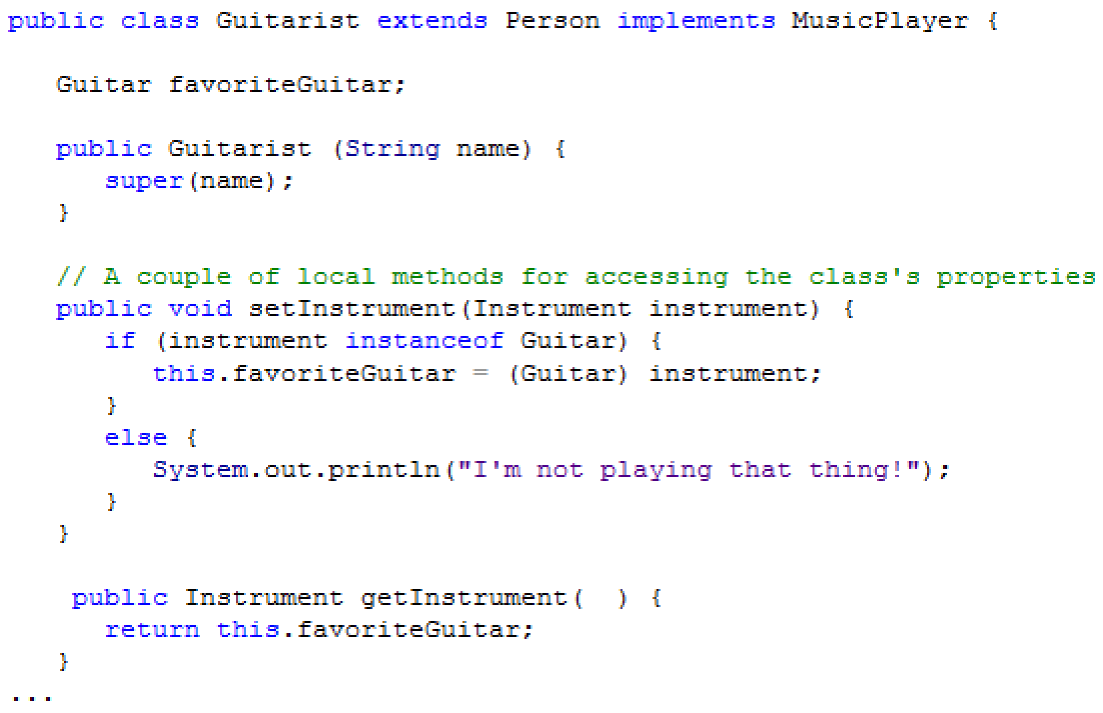

Natural Language Description:
The system needs to be large, with four legs and a trunk
=
Natural Language Description:
The system needs to be large, with four legs and a trunk
=
Communication + Ambiguity = Confusion!
✕
The System Designer's Perspective
The System Implementer's Perspective

"**The system needs to be large,**" an elephant is **large**.

"**with four legs**" the elephant has 4 legs.

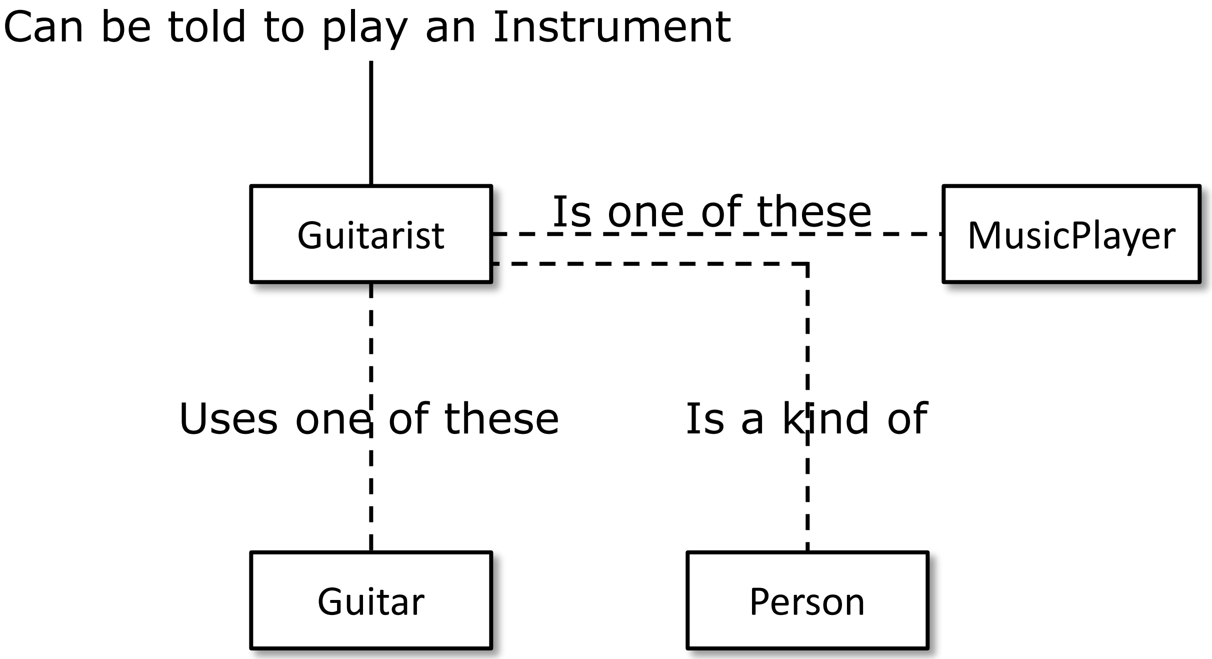
"**and a trunk**" the elephant a body.

1.4. Detailed Overload: Modeling with Code

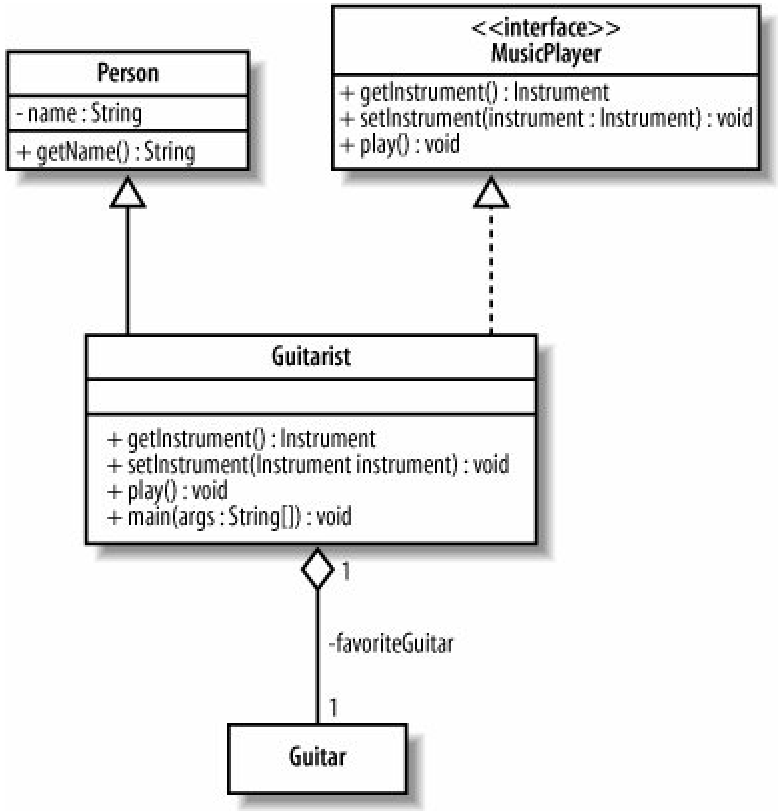


```java
public class Guitarist extends Person implements MusicPlayer {

   Guitar favoriteGuitar;

   public Guitarist (String name) {
      super(name);
   }

   // A couple of local methods for accessing the class's properties
   public void setInstrument(Instrument instrument) {
      if (instrument instanceof Guitar) {
         this.favoriteGuitar = (Guitar) instrument;
      }
      else {
         System.out.println("I'm not playing that thing!");
      }
   }

    public Instrument getInstrument(  ) {
      return this.favoriteGuitar;
   }
...
```

1.4. Detailed Overload: Modeling with Code



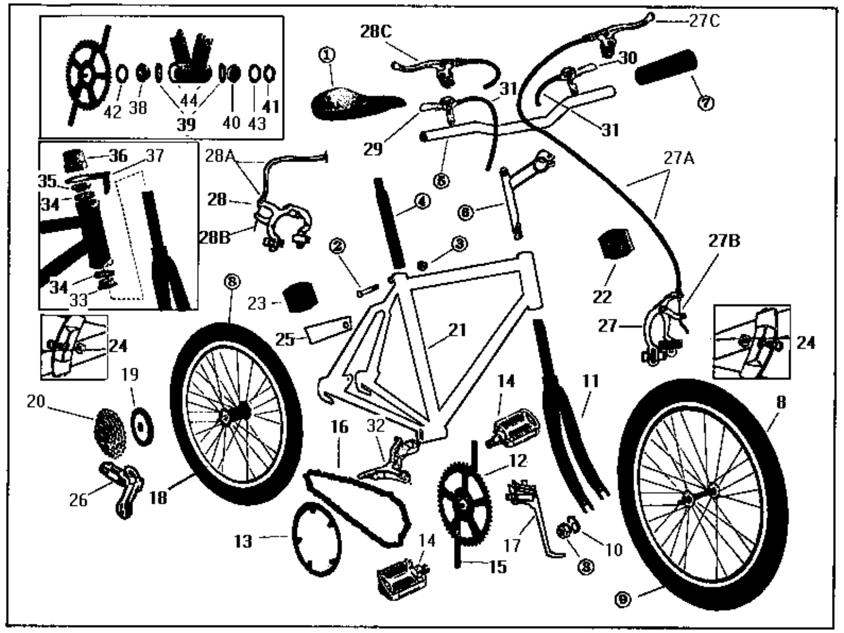
1.5. Getting the Balance Right: Formal Languages


Person
- name : String
+ getName() : String
<<interface>>
MusicPlayer
+ getInstrument() : Instrument
+ setInstrument(instrument : Instrument) : void
+ play() : void
Guitarist
+ getInstrument() : Instrument
+ setInstrument(Instrument instrument) : void
+ play() : void
+ main(args : String[]) : void
1
-favoriteGuitar
1
Guitar

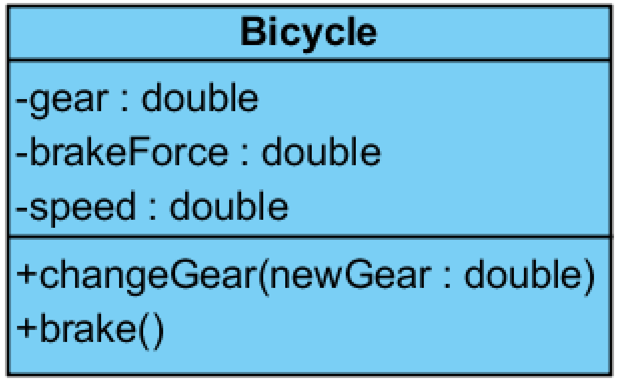
2. Models and Diagrams

- UML modeling is not just about diagrams
- A particular diagram will show you some parts of your model but not everything
- A model is made up by a collection of elements including their connections to each other
- Diagrams come in to play when we need to create new elements or to organize related elements

Examples:



Model



Bicycle

-gear : double
-brakeForce : double
-speed : double

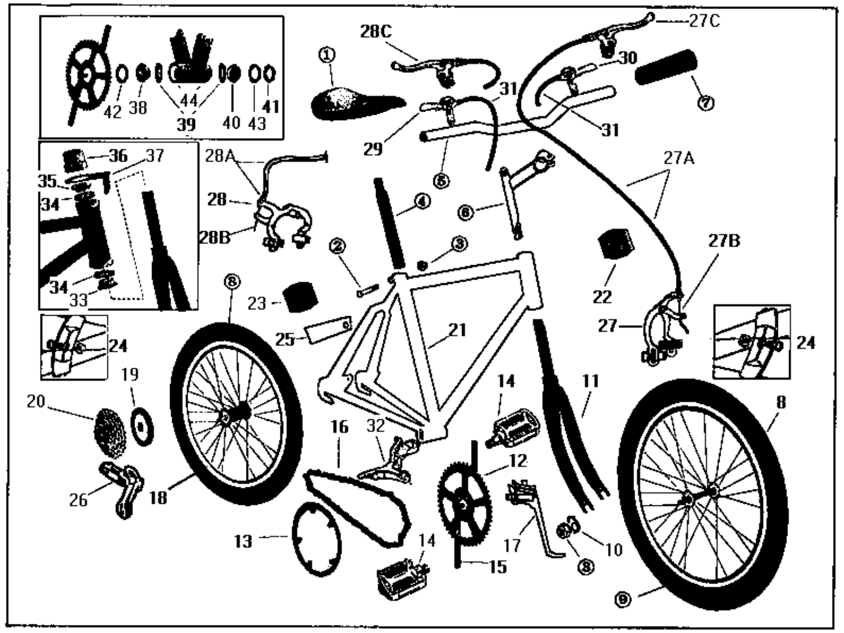
+changeGear(newGear : double)
+brake()

Diagram

2.1. Model in Big Picture

- Model is a thing used as example to follow or imitate.
- Model sits behind modeling tool
- Some models are quite big that can be separated into small parts to make detail representation.

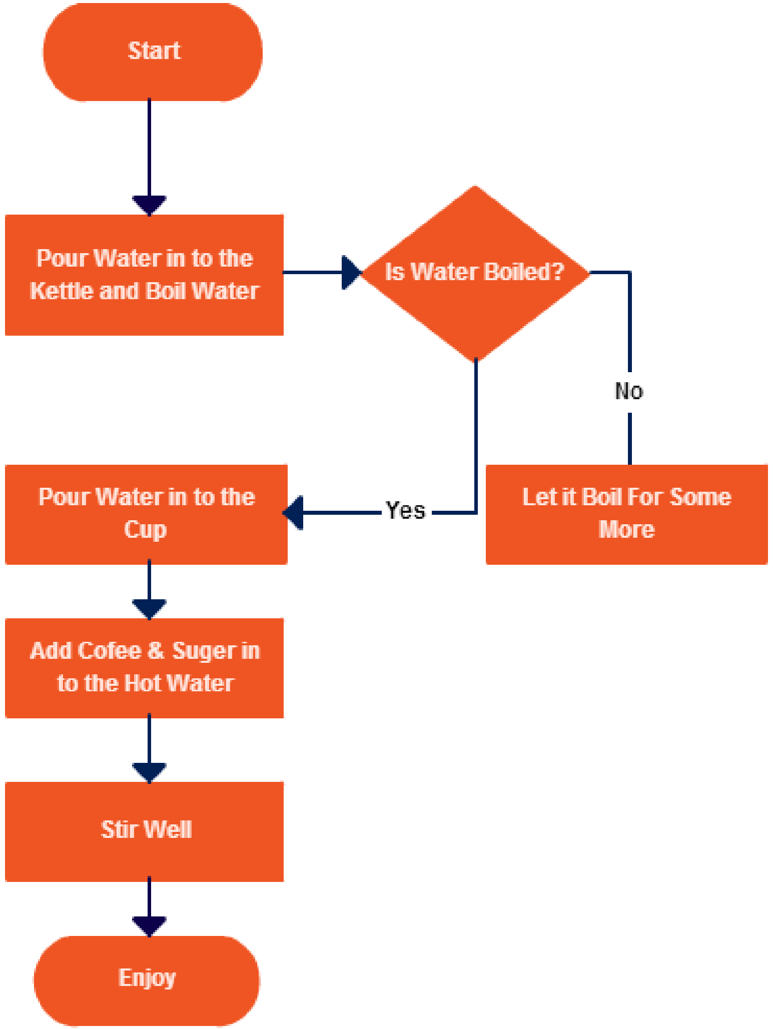
Examples:



2.2. Roles of Diagram

- Diagram a simplified drawing showing the appearance, structure, or workings of something; a schematic representation.
- Diagram makes a system easily understand
- Diagram explains how a system work
- Diagram shows structure of a system.

Example:

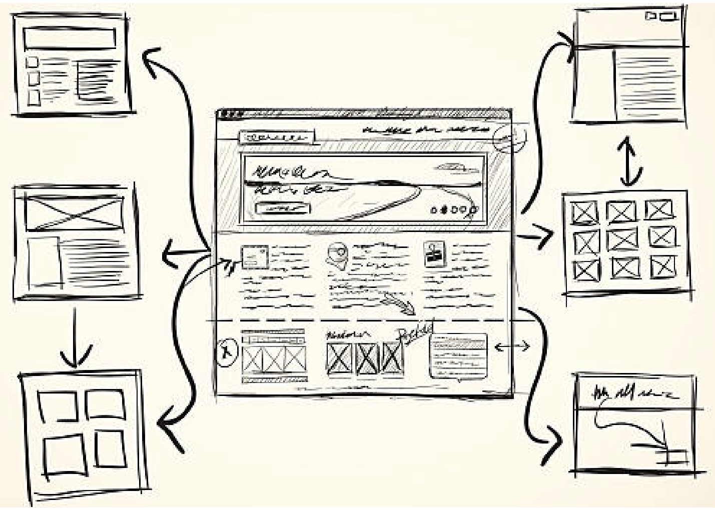

Start
Pour Water in to the Kettle and Boil Water
Is Water Boiled?
No
Let it Boil For Some More
Yes
Pour Water in to the Cup
Add Cofee & Suger in to the Hot Water
Stir Well
Enjoy

2.3. Relation of Models and Diagrams

- Diagram displays overall image of a model
- Diagram explains in details about a model
- Model sits behind diagrams
- Diagram may show a part of a model.

Example:

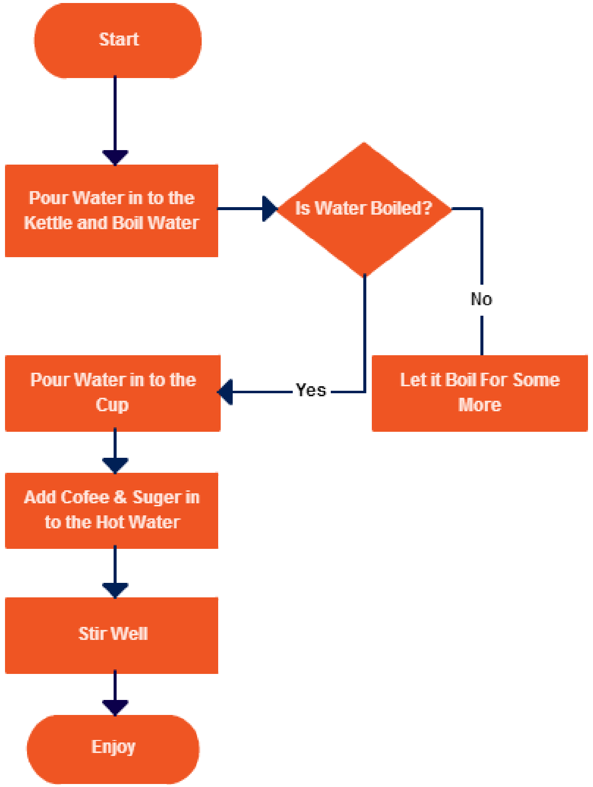
3. Degree of UML



UML as a sketch

```
public class Bicycle{
  private double gear;
  private double brakeForce;
  private double speed;
  public void changeGear(double newGear){
     gear = newGear;
  }
  public void brake(){
  }
}
```

UML as a programming language

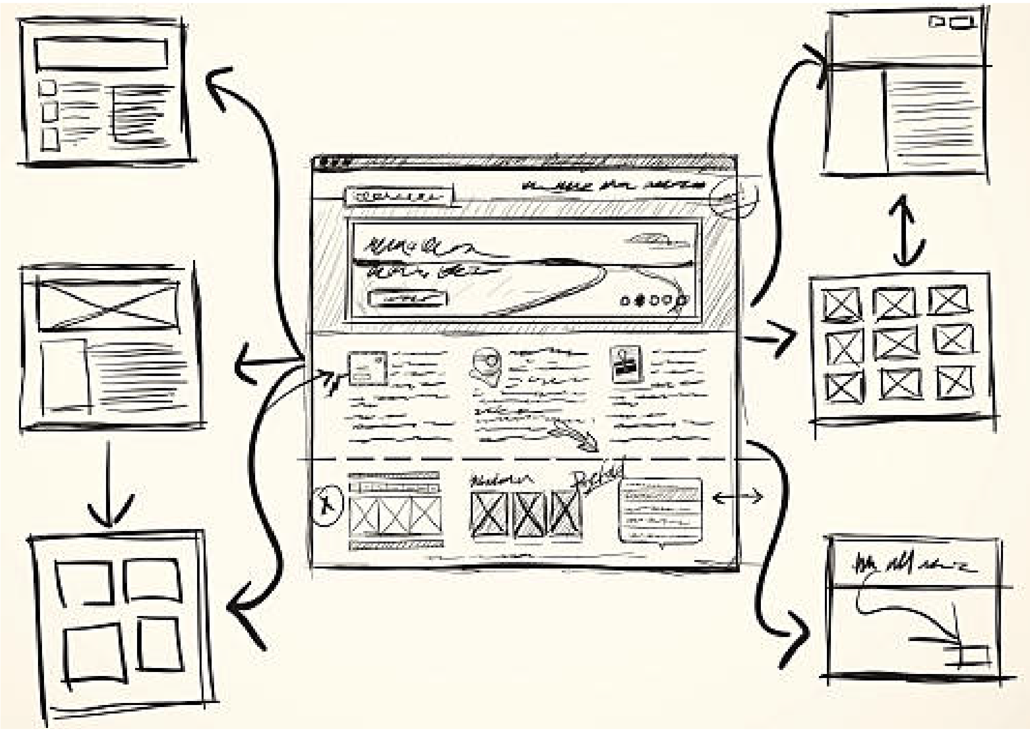

Start
Pour Water in to the Kettle and Boil Water
Is Water Boiled?
No
Let it Boil For Some More
Yes
Pour Water in to the Cup
Add Cofee & Suger in to the Hot Water
Stir Well
Enjoy

UML as a blueprint

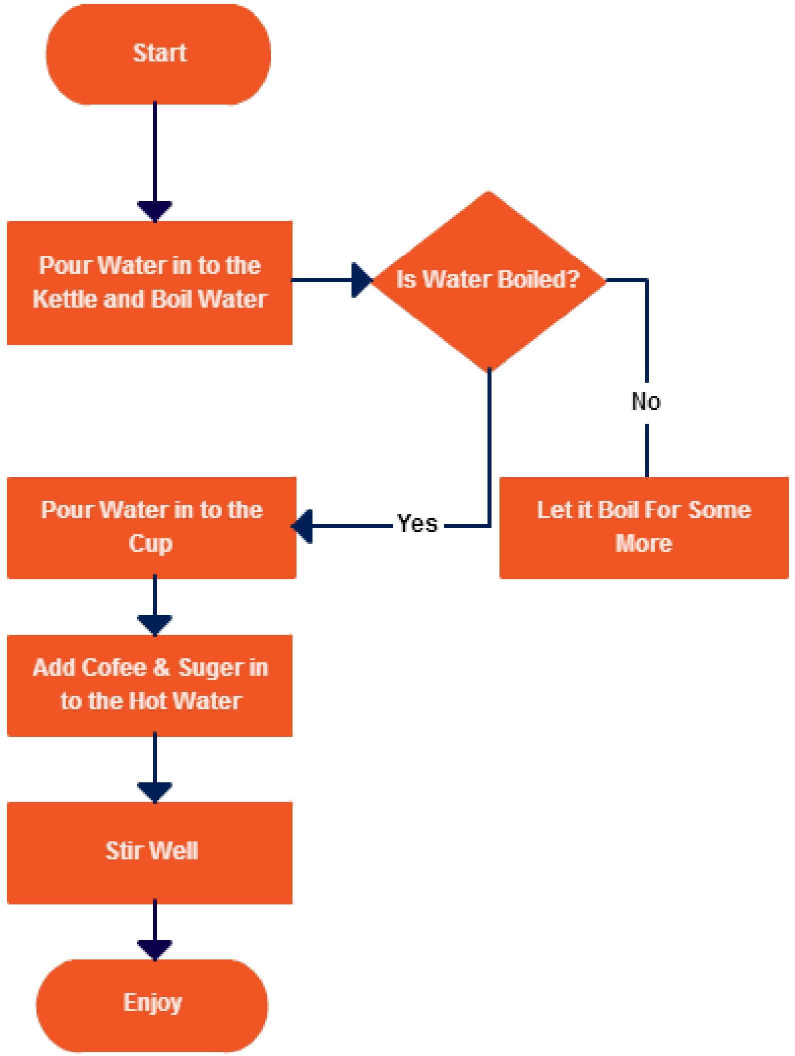
3.1. UML as a Sketch

- Use UML to make brief sketches to convey key points.
- These are throwaway sketches
- They could be written on a whiteboard or paper.

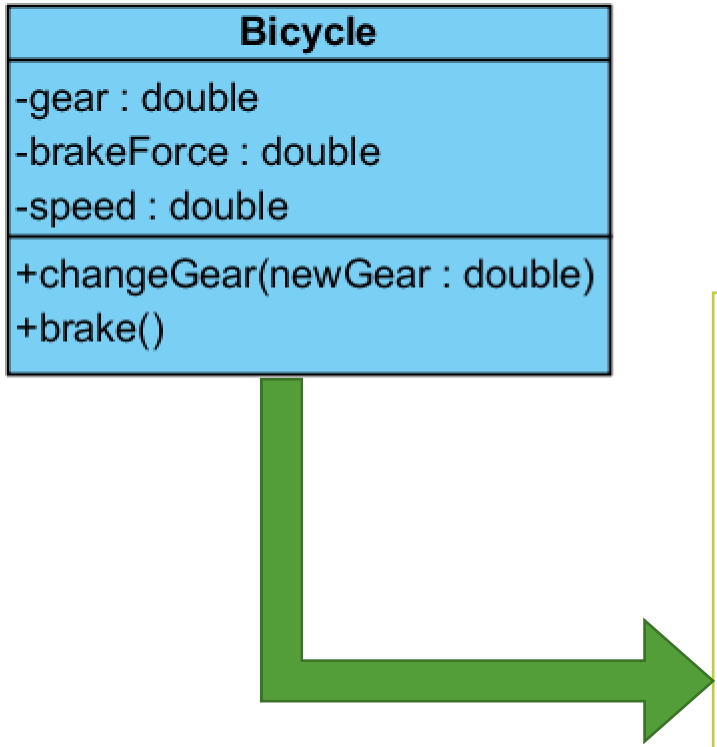
Examples:



3.2. UML as Blueprint


Start
Pour Water in to the Kettle and Boil Water
Is Water Boiled?
No
Let it Boil For Some More
Yes
Pour Water in to the Cup
Add Cofee & Suger in to the Hot Water
Stir Well
Enjoy

3.3. UML as a Programming Language



Bicycle

-gear : double
-brakeForce : double
-speed : double

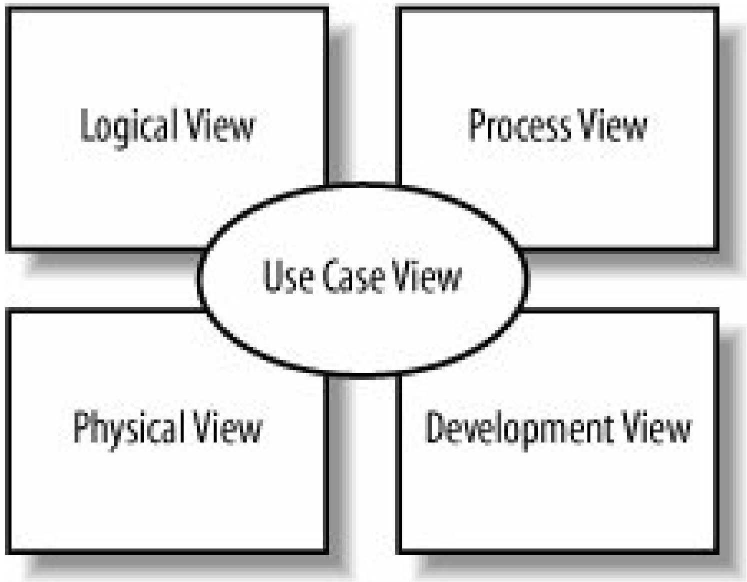
+changeGear(newGear : double)
+brake()

```
public class Bicycle{
  private double gear;
  private double brakeForce;
  private double speed;
  public void changeGear(double newGear){
     gear = newGear;
  }
  public void brake(){
  }
}
```

Generate code from UML diagram.

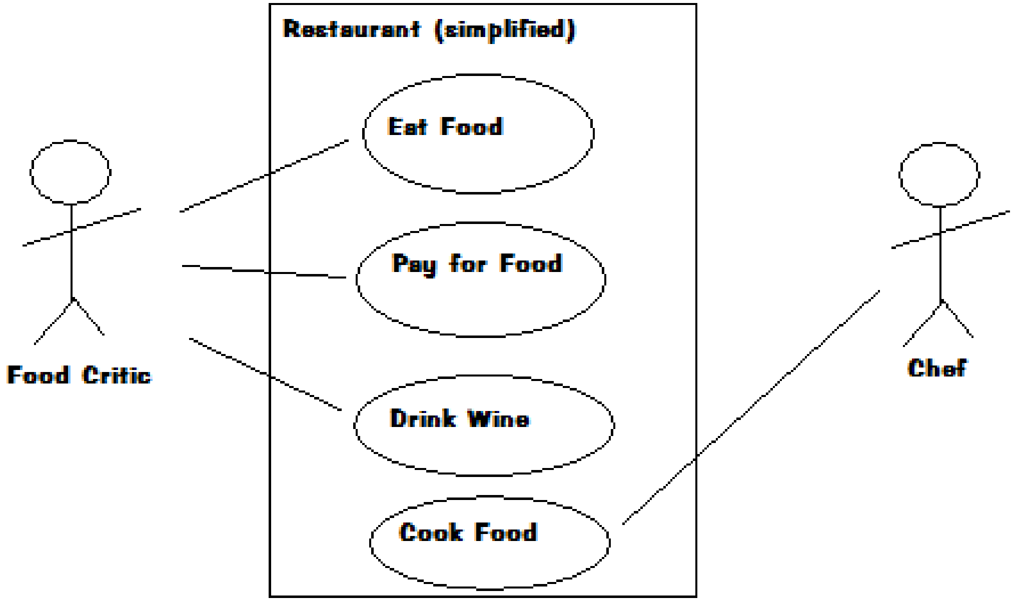
4. Views of Your Model

- The UML model can be broken down into perspectives or views to capture a particular facet of your system
- Kruchten's 4+1 view model
  - Logical view
  - Process view
  - Development view
  - Physical view
  - Use case view



4.1. Use Case View

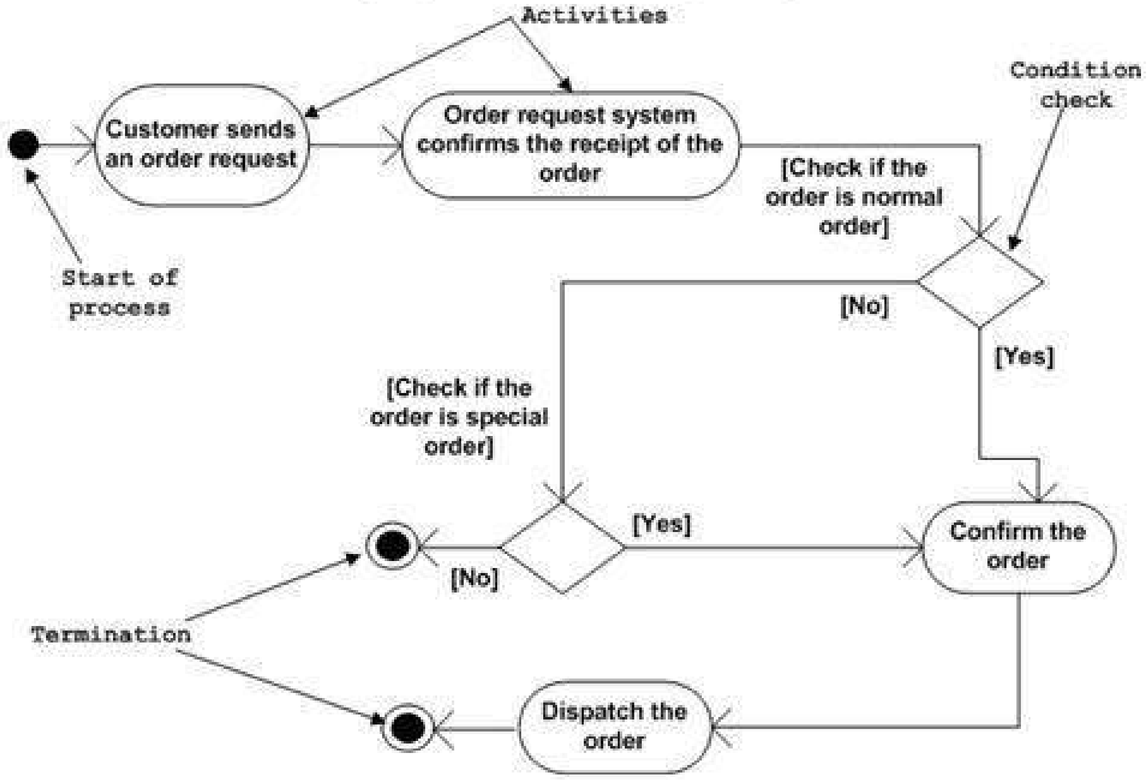
- Use cases are starting point
- Use cases specify only what the system is supposed to do, not what it shouldn't do
- It is important for system designer
- Priority and risk can be assigned to each use case
- Use cases help construct tests for your system



4.2. Process View

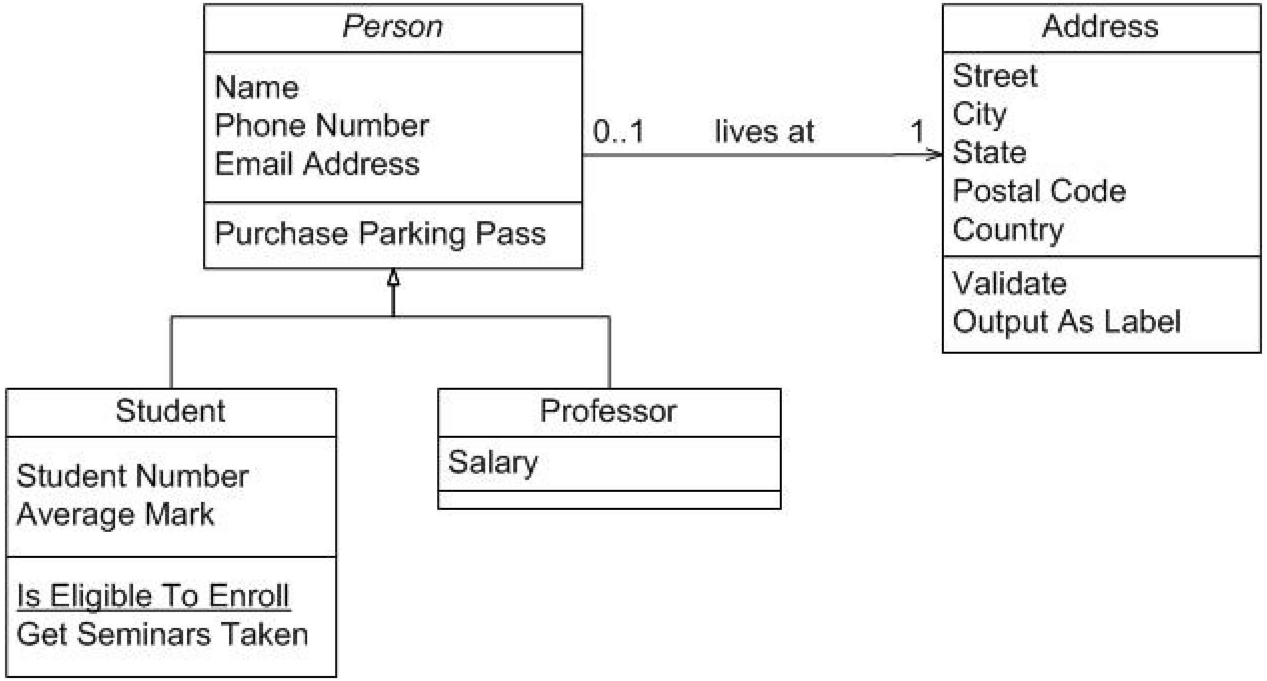
- Process view has only one diagram is Activity diagram
- Allow you to specify how your system will accomplish its goals
- Modeling business processes

Activity diagram of an order management system


Activities
Condition check
Customer sends an order request
Order request system confirms the receipt of the order
[Check if the order is normal order]
Start of process
[No]
[Yes]
[Check if the order is special order]
[Yes]
[No]
Confirm the order
Termination
Dispatch the order

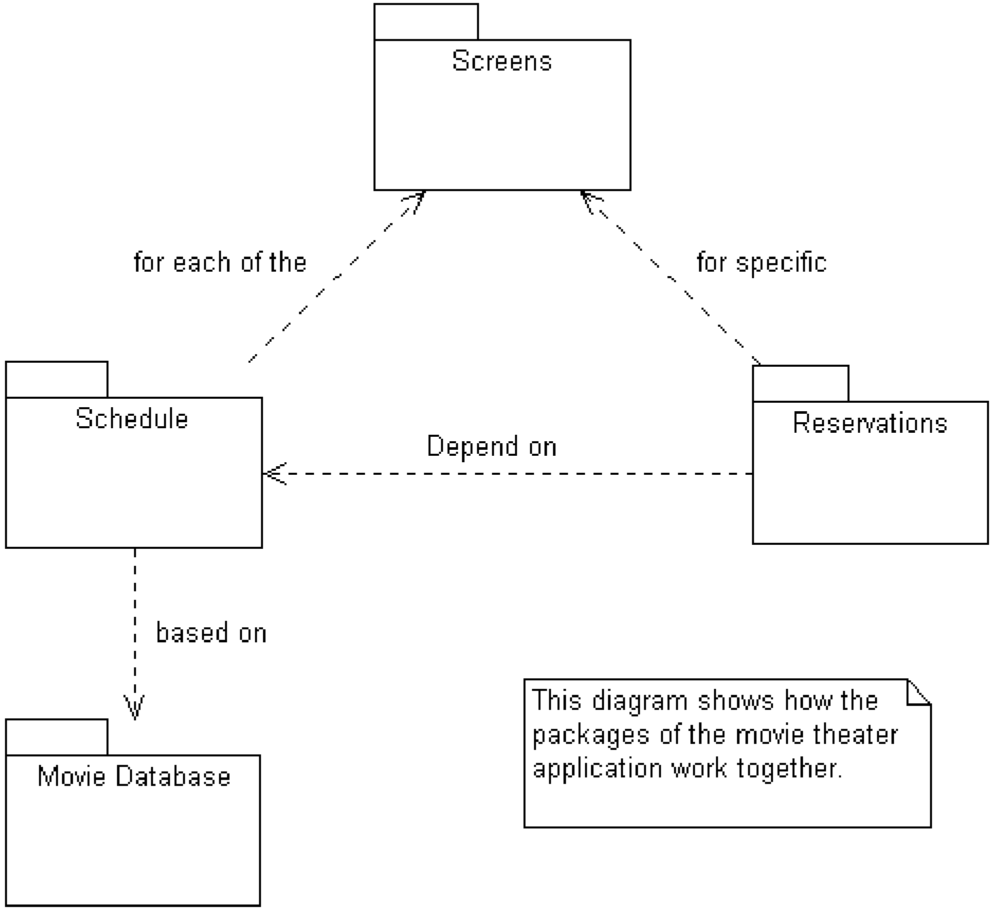
4.3. Logical View

- Classes are at the heart of any object oriented system
- The system's structure is made up of a collection of pieces referred to as objects
- Classes describe the different types of objects



4.4. Development View

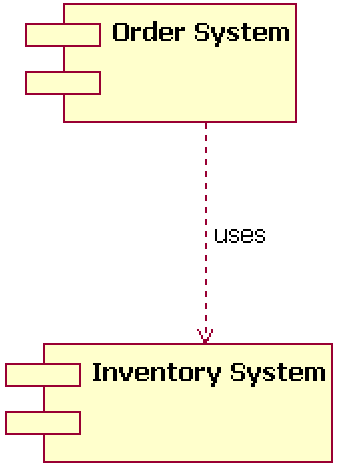
- The development view illustrates a system from a programmer's perspective
- It is concerned with software management
- This view is also known as the implementation view
- It uses the UML Component diagram to describe system components
- Package diagram is used to describe Development View



This diagram shows how the packages of the movie theater application work together.

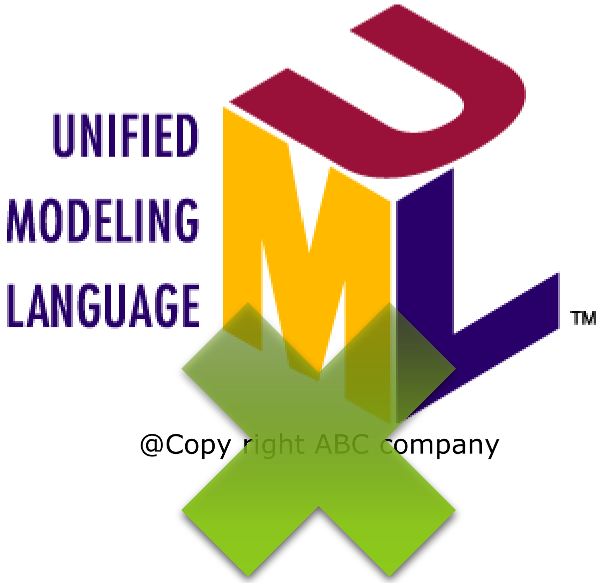
4.5. Physical View

- The process of Deployment of the system to end user
- The Deployment diagram shows the process of deployment
- The Deployment process can include:
  - Setting up and Installation
  - Maintenance



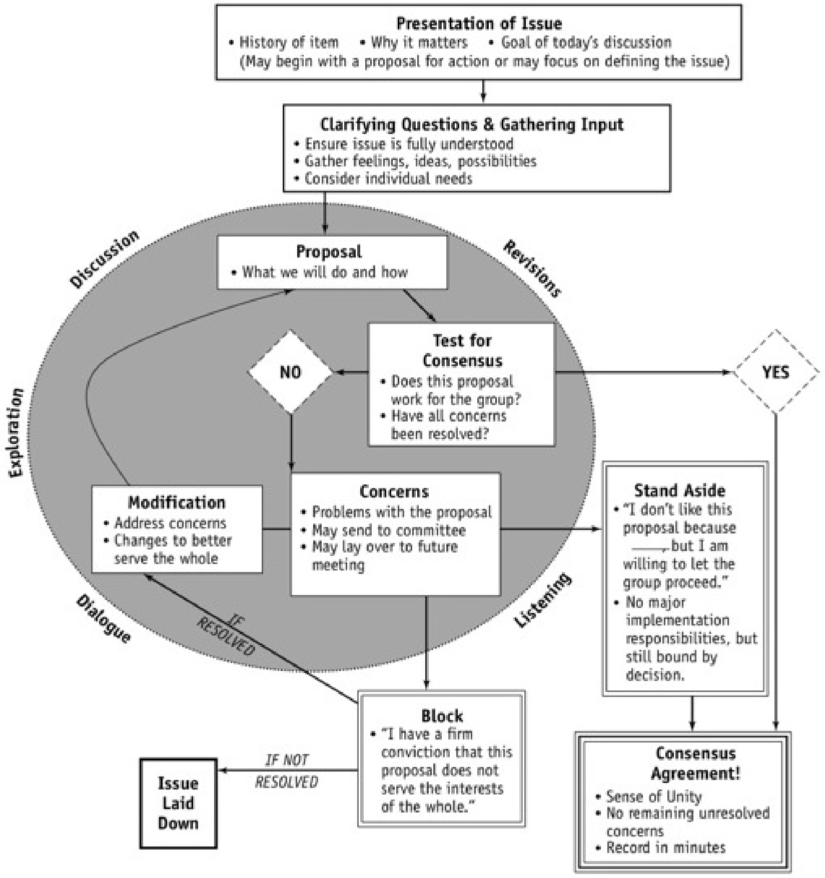
5. Dispelling Misconceptions about UML

- UML is not proprietary (Not copyrighted)
- UML is not difficult
- UML is not time-consuming



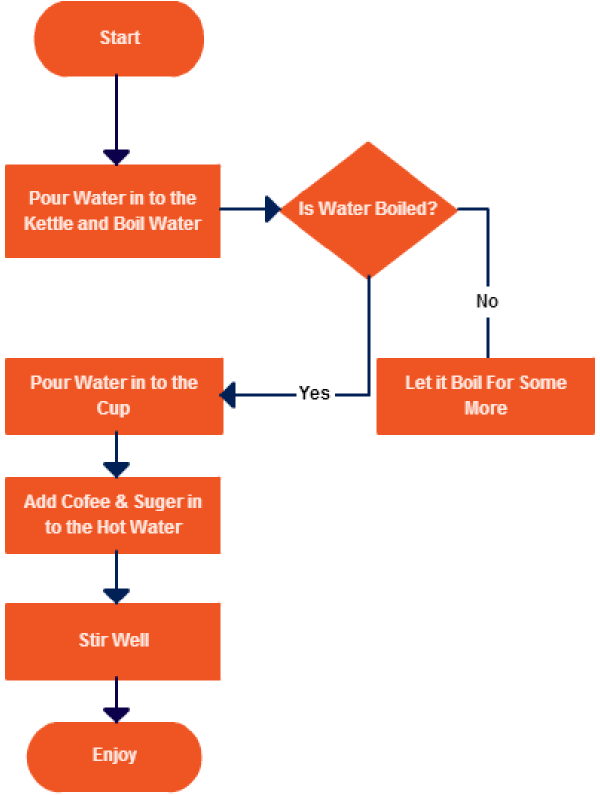
5.1. When to draw diagrams

- When several people need to understand the structure of a particular part of the design
- When two or more people disagree on how a particular element should be designed, and you want team consensus
- When you just want to play with a design idea, and the diagrams can help you think it through


Presentation of Issue
• History of item • Why it matters • Goal of today's discussion
(May begin with a proposal for action or may focus on defining the issue)
Clarifying Questions & Gathering Input
• Ensure issue is fully understood
• Gather feelings, ideas, possibilities
• Consider individual needs
Discussion
Revisions
Exploration
Dialogue
Listening
Proposal
• What we will do and how
Test for Consensus
• Does this proposal work for the group?
• Have all concerns been resolved?
NO
YES
Modification
• Address concerns
• Changes to better serve the whole
Concerns
• Problems with the proposal
• May send to committee
• May lay over to future meeting
Stand Aside
• "I don't like this proposal because ____, but I am willing to let the group proceed."
• No major implementation responsibilities, but still bound by decision.
IF RESOLVED
Block
• "I have a firm conviction that this proposal does not serve the interests of the whole."
IF NOT RESOLVED
Issue Laid Down
Consensus Agreement!
• Sense of Unity
• No remaining unresolved concerns
• Record in minutes

5.2. When NOT to draw diagrams

- Don’t draw diagrams because the process tells you to
- Don’t draw diagrams because you feel guilty not drawing them
- Don’t draw diagrams to create comprehensive documentation of the design phase prior to coding
- Don’t draw diagrams for other people to code



Start

Pour Water in to the Kettle and Boil Water

Is Water Boiled?

No

Let it Boil For Some More

Yes

Pour Water in to the Cup

Add Cofee & Suger in to the Hot Water

Stir Well

Enjoy

5.3. Example of a medium project

12 people working on a project of a million lines of Java will require 25 to 200 pages of persistent documentation that include:

- UML diagrams of the high level structure of the important modules
- ER diagrams of the relational schema
- A page or two about how to build the system
- Testing instructions
- Source code control instructions
- etc.

Test

Question	Possible answers	Correct Answer
1. Choose three advantages of UML:	a) It's concise b) It's runnable c) It's the standard d) It's formal language	a) It's concise c) It's the standard d) It's formal language
2. Completing blank field:	A model is made up by a ……… ……………including their ………… ……………………………………………..to each other	collection of elements their connections
3. Choose a name that is not in the 4+1 View Model:	a) Logical view b) Use Case view c) Process view d) Seaside view e) Physical view f) Development view	d) Seaside view

Test

Question	Possible answers	Correct Answer
4. Activity Diagram is used to represent:	a) Logical view b) Use Case view c) Process view d) Physical view e) Development view	c) Process view
5. Completing blank field:	UML is not proprietary, UML is not difficult, UML is ………………………	Not time-consuming

Summarize

- UML contains diagrams that describe models of a program.
- Use Informal language will mostly lead to confuses, whereas, UML as formal language that omit confusing and everyone know it.
- Diagram contains elements and its relationships that model the program.
- UML is used to make sketch, blueprint, and programming language.
- UML is broken down into 4+1 view model such as Logical, Process, Development, Physical, and Use Case view.
- UML is simple, not proprietary, and not time-consuming

References

- Miles, R. (2006). Learning UML 2.0. O’Reilly
- Chonoles, M. & Schardt, J. (2003). UML 2 for Dummies. Wiley Publishing

Next Lesson

Use Case Diagram

1. Use Case components
2. Use Case relationships
3. UML drawing tools
4. StarUML
5. Examples