

Architectural Structural Design

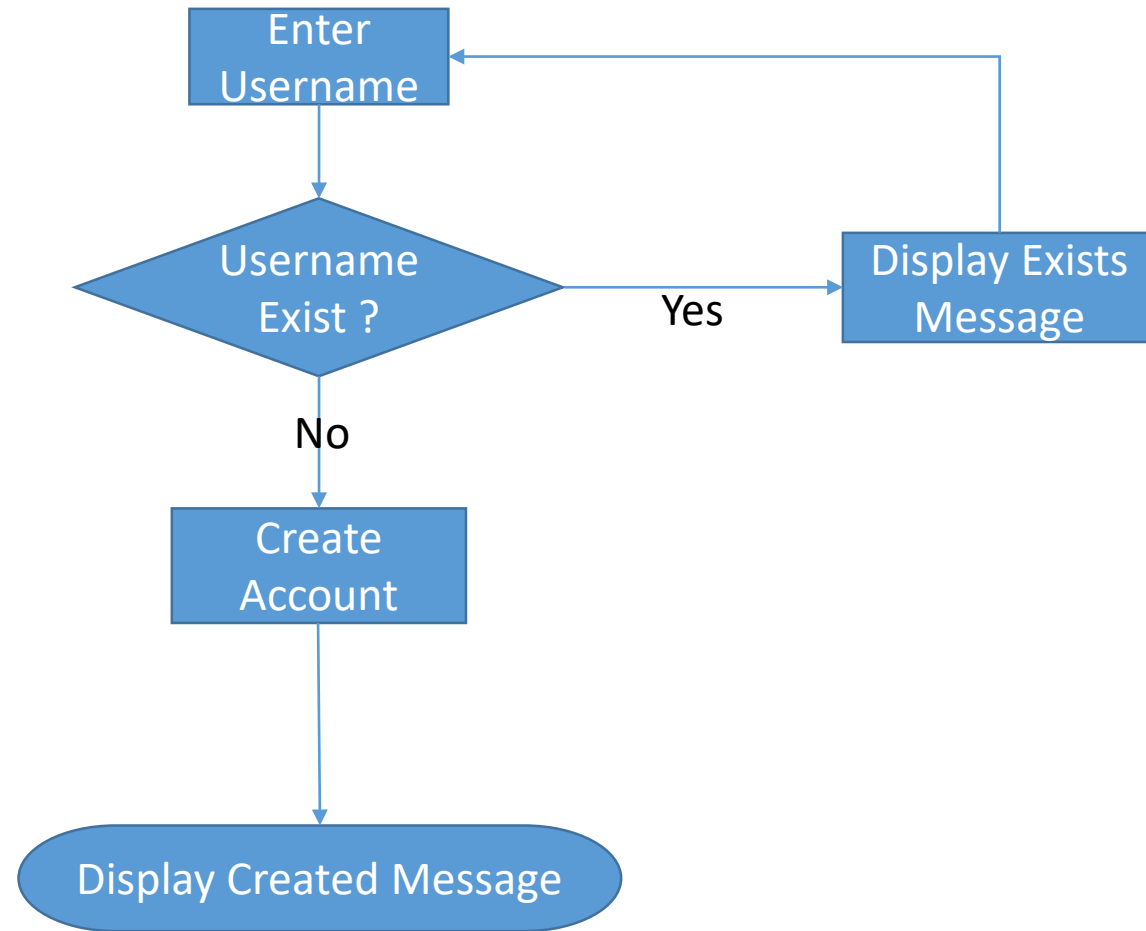
I have gathered my requirements, now what?

- Design stage
 - Remember the SDLC
- How do I design ?
 - Data flow diagrams
 - UML
 - Descriptions
 - Scribbles

Flowcharts

- Graphical representation of a process
- Provides chronology and flow
- Indicates path deviation in a process

Adding a new user process

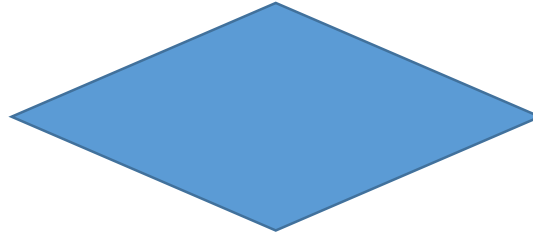


Common Components

- Process



- Decision

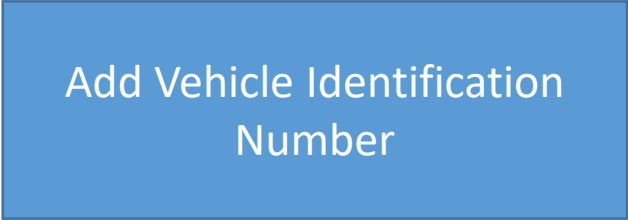


- Alternate Process



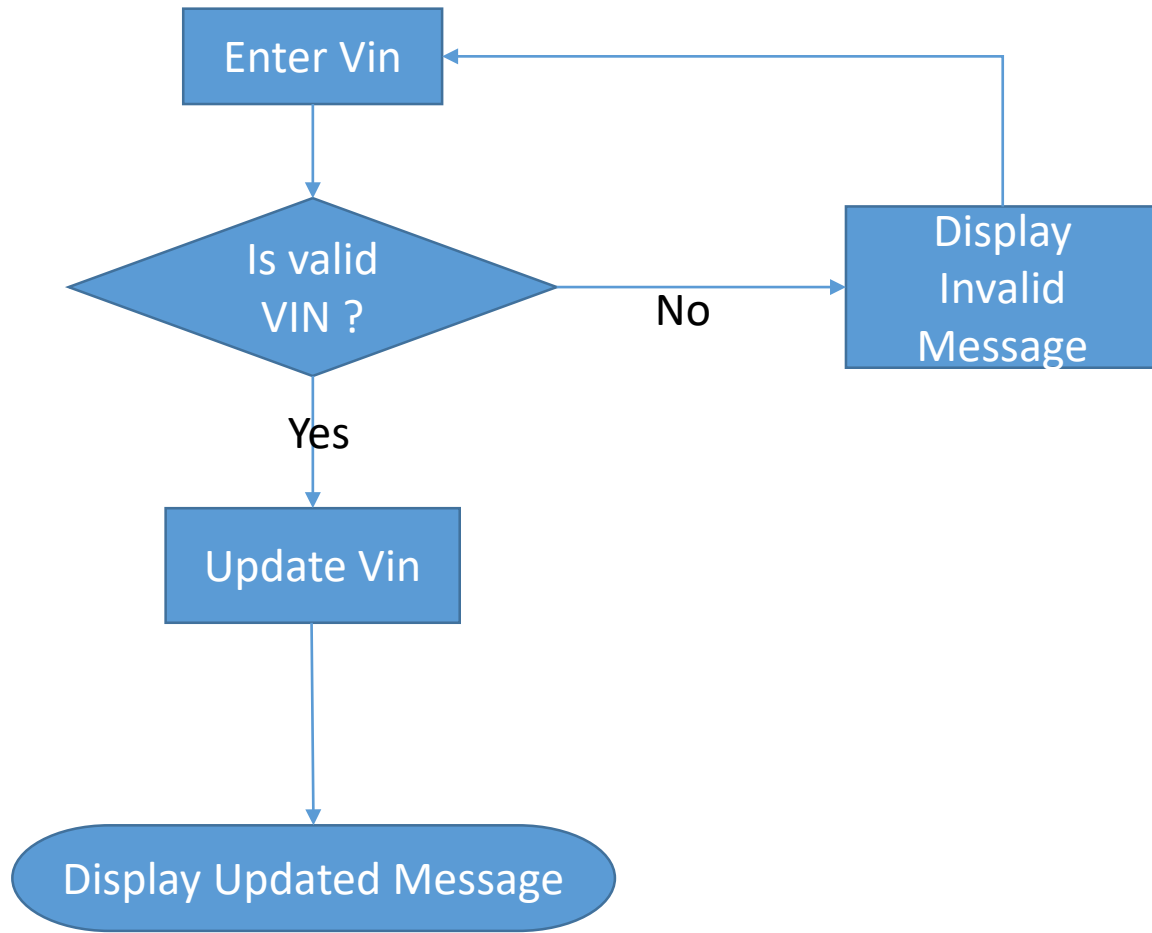
Process

- Fundamental driver of a flowchart
- Symbolises a process taking place



Add Vehicle Identification
Number

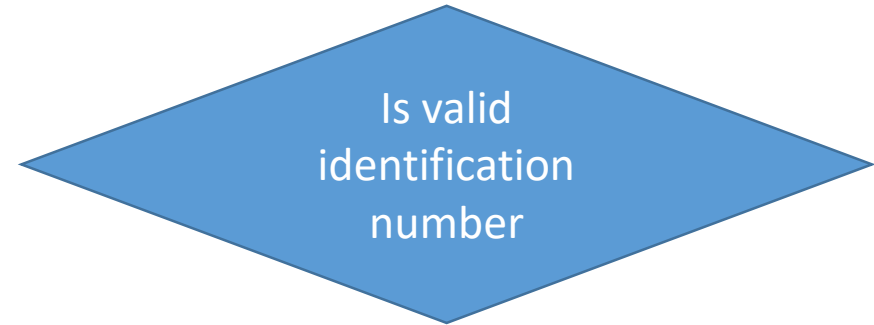
Process Control and Code



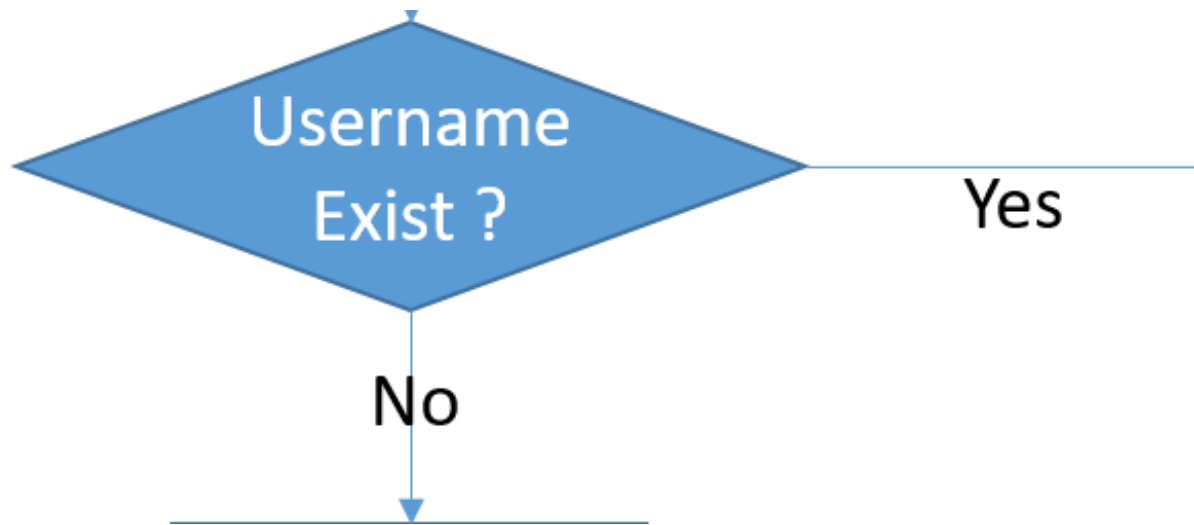
```
void AddVin(Vehicle* vehicle, char* vin)
{
    vehicle->UpdateVin(vin);
}
```

Conditional Branching

- Process of making decisions
- Decisions are a fundamental part of software processes
- Provides means by which control or data may take differing paths through the software to achieve differing goals



Branching in Code

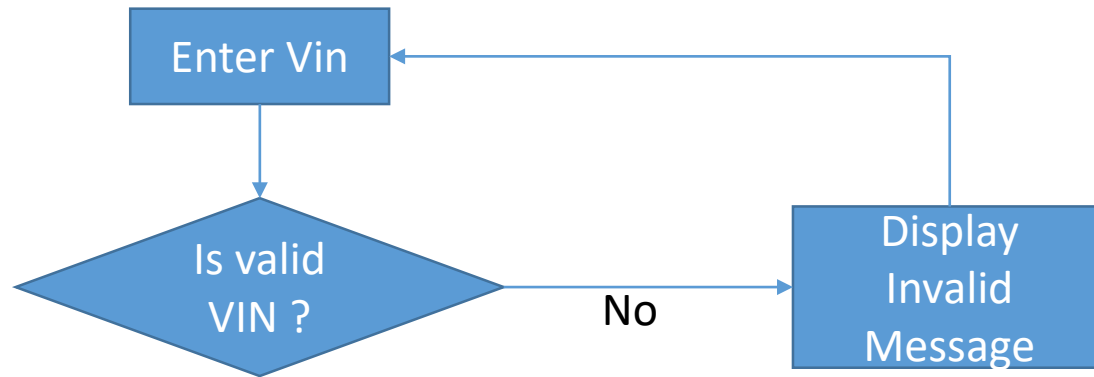


```
void AddVin(Vehicle* vehicle, char* vin)
{
    if (sizeof(vin) == 17)
    {
        vehicle->UpdateVin(vin);
    }
    else
    {
        printf("Error:%d", errorCode);
    }
}
```

Iteration

- Fundamental component of software
- Allows for continued execution of given operations
- Can be terminated with conditional values

Iteration in Code



```
void AddVin(Vehicle* vehicle, char*  
vin)  
{  
  
while (!ValidVin(vin))  
{  
    printf("Invalid Vin");  
}  
}
```

Logic and Operation

- Flowcharts provide for a logical flow of operations in the system
- Reflective of the algorithmic decomposition approach
- Primitive in nature, does not necessarily represent all elements of the system

From picture to prose ?

- Pictures are not the only way to provide a system design
- Architectural description language (ADL) can be used to denote process transition in a system
- Defined formal language for system representation

Description (ADL)

- AADE – Architecture Analysis and Design Language
- ACME - The Architectural Based Language and Environment
(resembles pseudocode)
- DARWIN – Graphical, provides representation of system in graphical form (hierarchical composition)
- AESOP – Akin to the UNIX model

ADL Examples

- ACME

```
Component pipe =  
{  
  Port in;  
  Port out;  
  Property implementation : String = "while (!in.eof)  
                                     { in.read; compute; out.write; }";  
}
```

Advantages of descriptions

- Descriptive linear account of system process flow
- Provides a complete picture of the flow of control through the system
- All operations can be accurately mapped

Disadvantages of descriptions

- Excessive text
- Hard to represent compartmentalisation
- OOP

Picture vs Text

- We can use pictures to design a system
- We can use text to represent a system
- Which is the more useful ?
 - An amalgamation of both
 - Enter UML

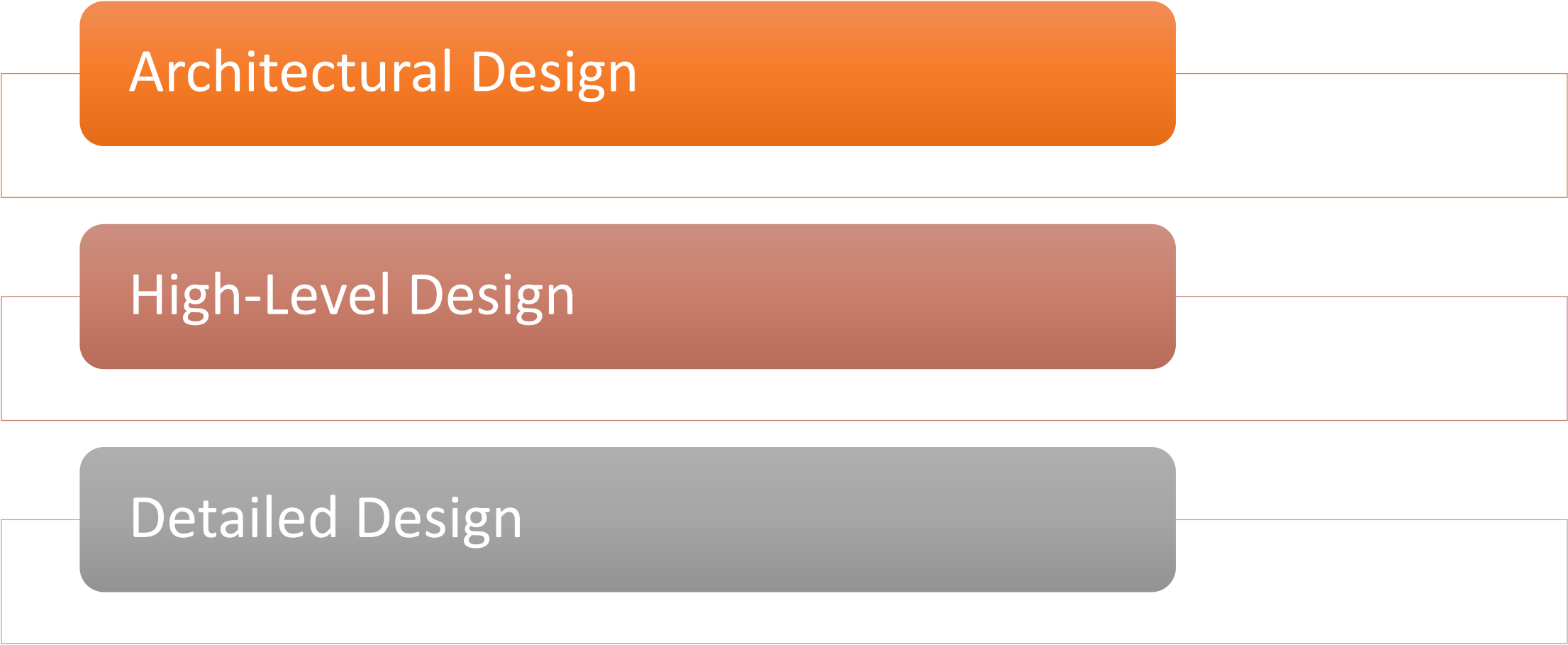
Task

- What other forms of ADL are there ?
- Describe at Least two of them
- What tools are available for them ?
- What other components can be found in flowcharts ?
- What are the purpose of them ?

UML

- Unified Modelling Language
 - Both graphical and textual representation of system design
- 3 Major revisions (1.0, 1.5, 2.1)
 - Includes minor revs periodically introducing specific diagrams

Levels of design



Architectural Design

High-Level Design

Detailed Design

Architectural Design

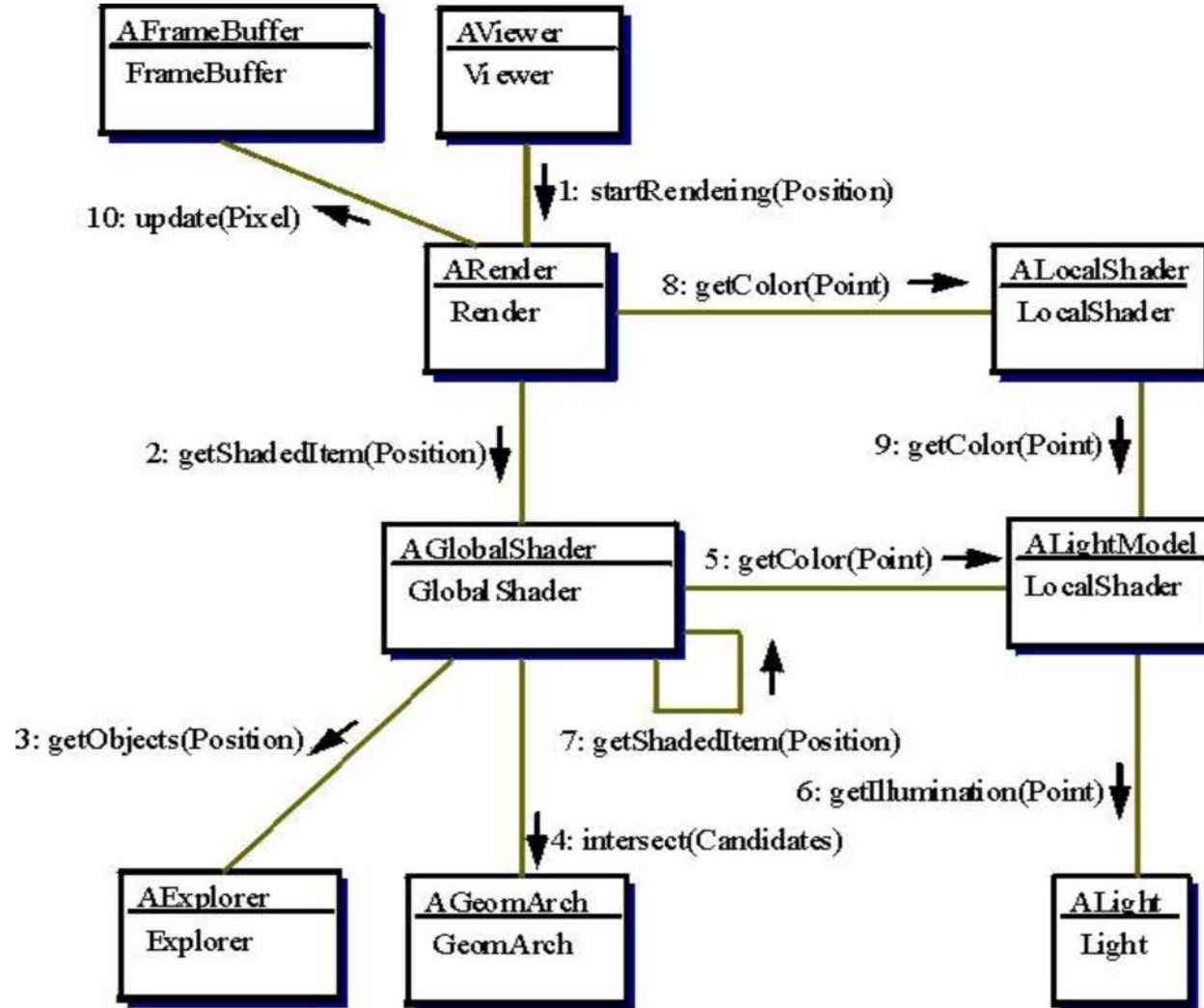
- Overview of the system includes :
 - Decomposition – Object Model
 - Compartmentalisation – Package Diagram
 - Class list and Diagram
 - Component Diagram
 - Deployment Diagram

Object Decomposition

- Compartmentalisation of the system into objects
- Objects represent physical entities
 - Record
 - Person
 - Vehicle
 - License

Object Model

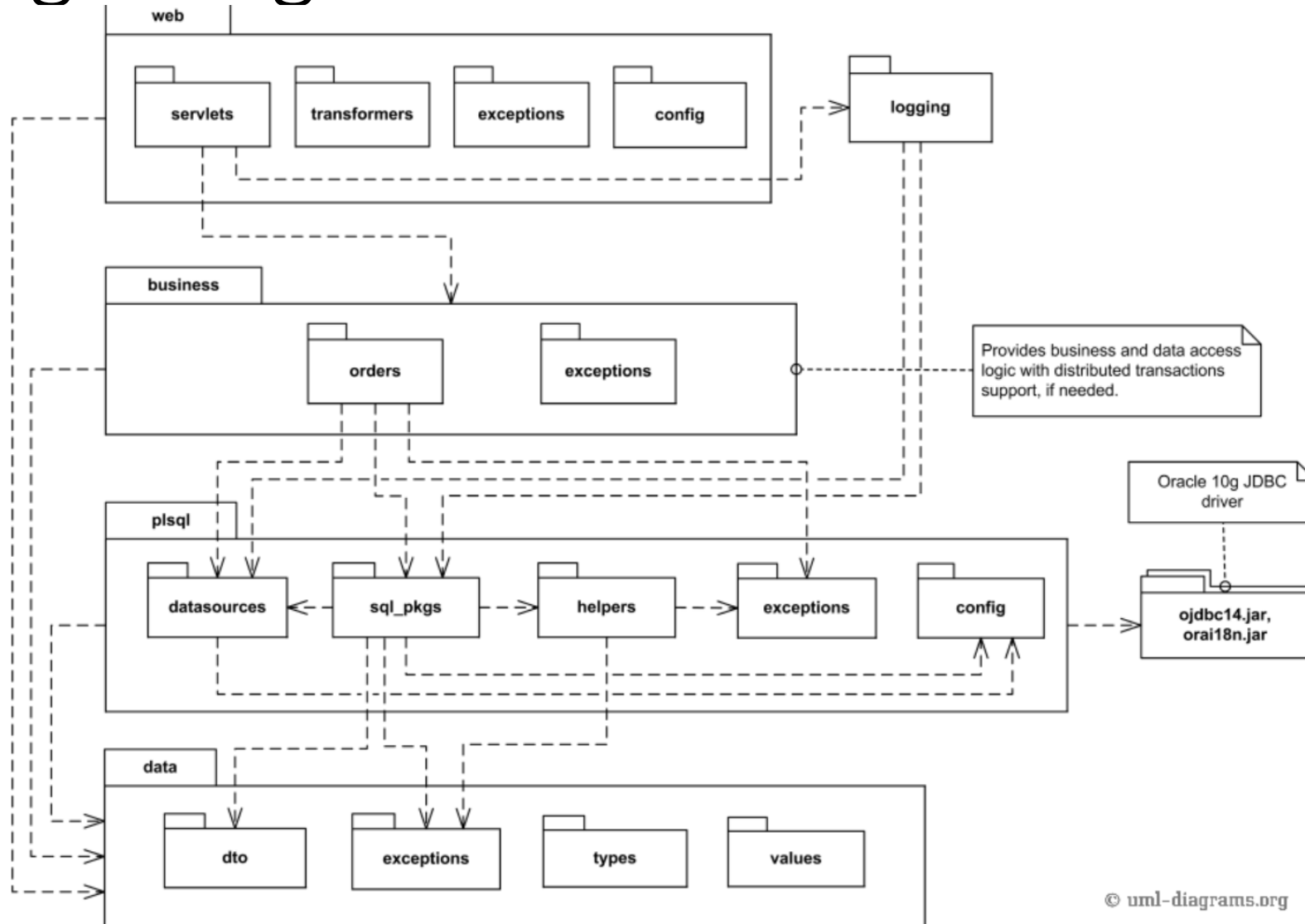
Association and Collaboration graphs of general approach of OO decomposition.



Modularisation / Packaging

- Compartmentalisation of the system
- System decomposed into modular units
- Provides an overview of the major modules within a system, inclusive of the components within.

Package Diagrams



Introduce a little class to your work

- Represent your system in smaller componential blocks
- Provides templates for the interactivity in the system
- Very simplistic in representation of the data structures in the system

The Diagram Basics

- A class is a template from which objects may be constructed

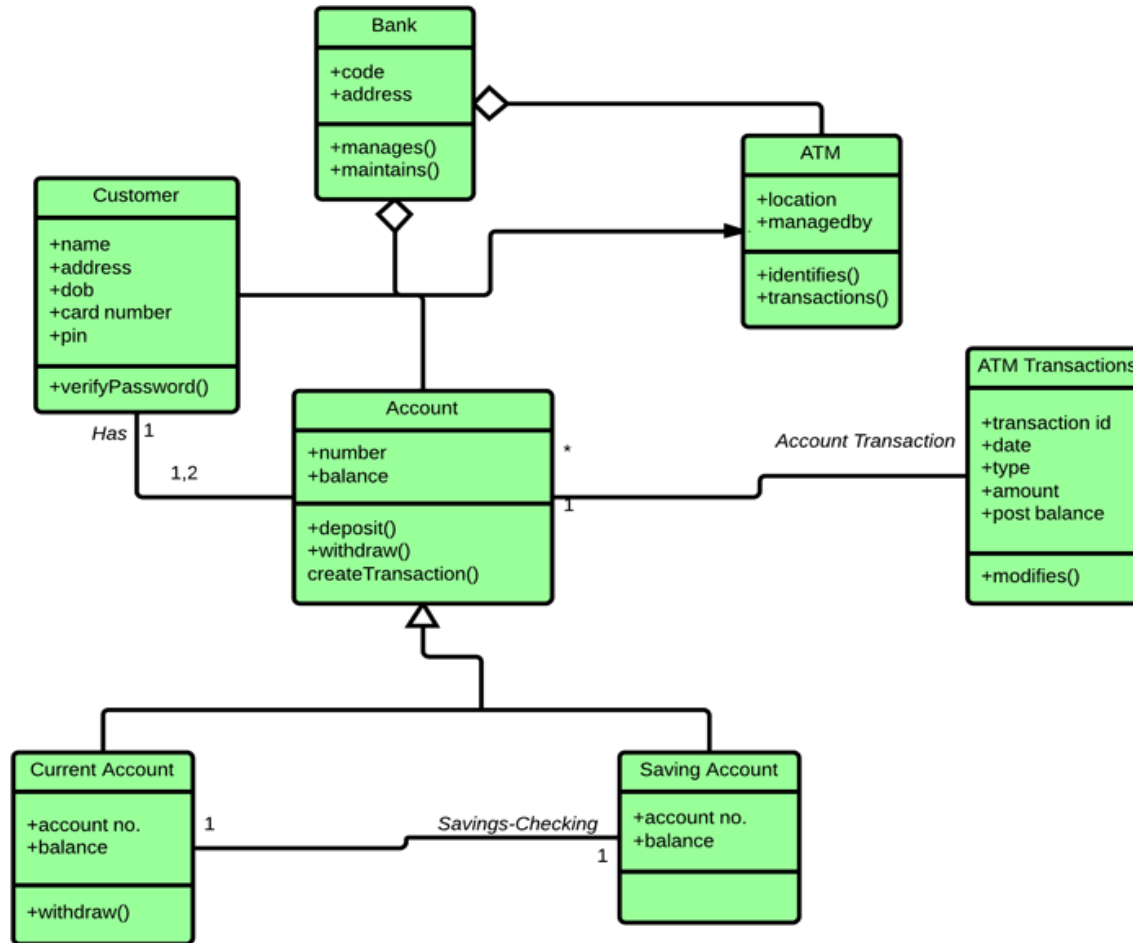


Class

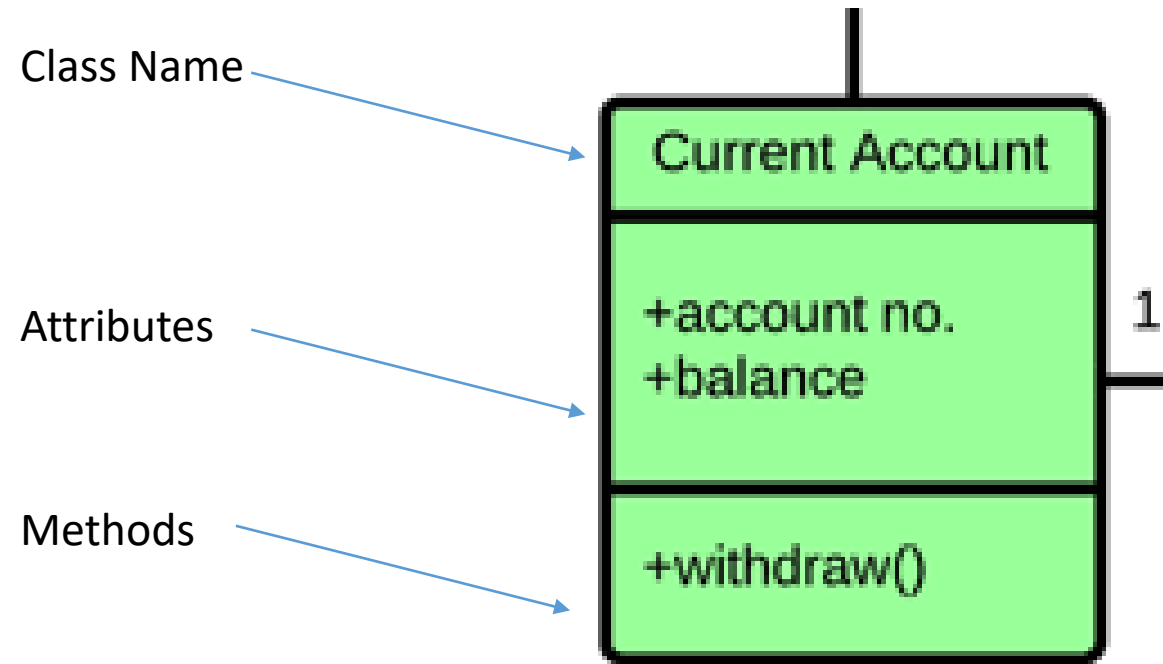


Object

Class Diagram



A Class



So what are the symbols for

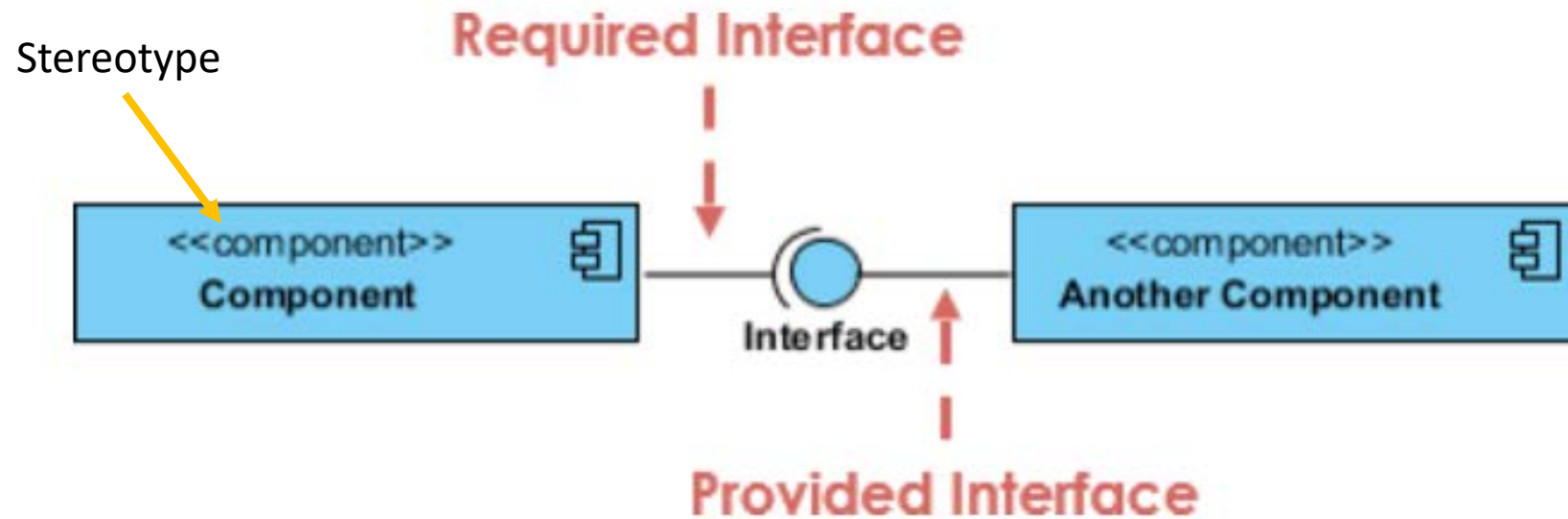
UML Notations:



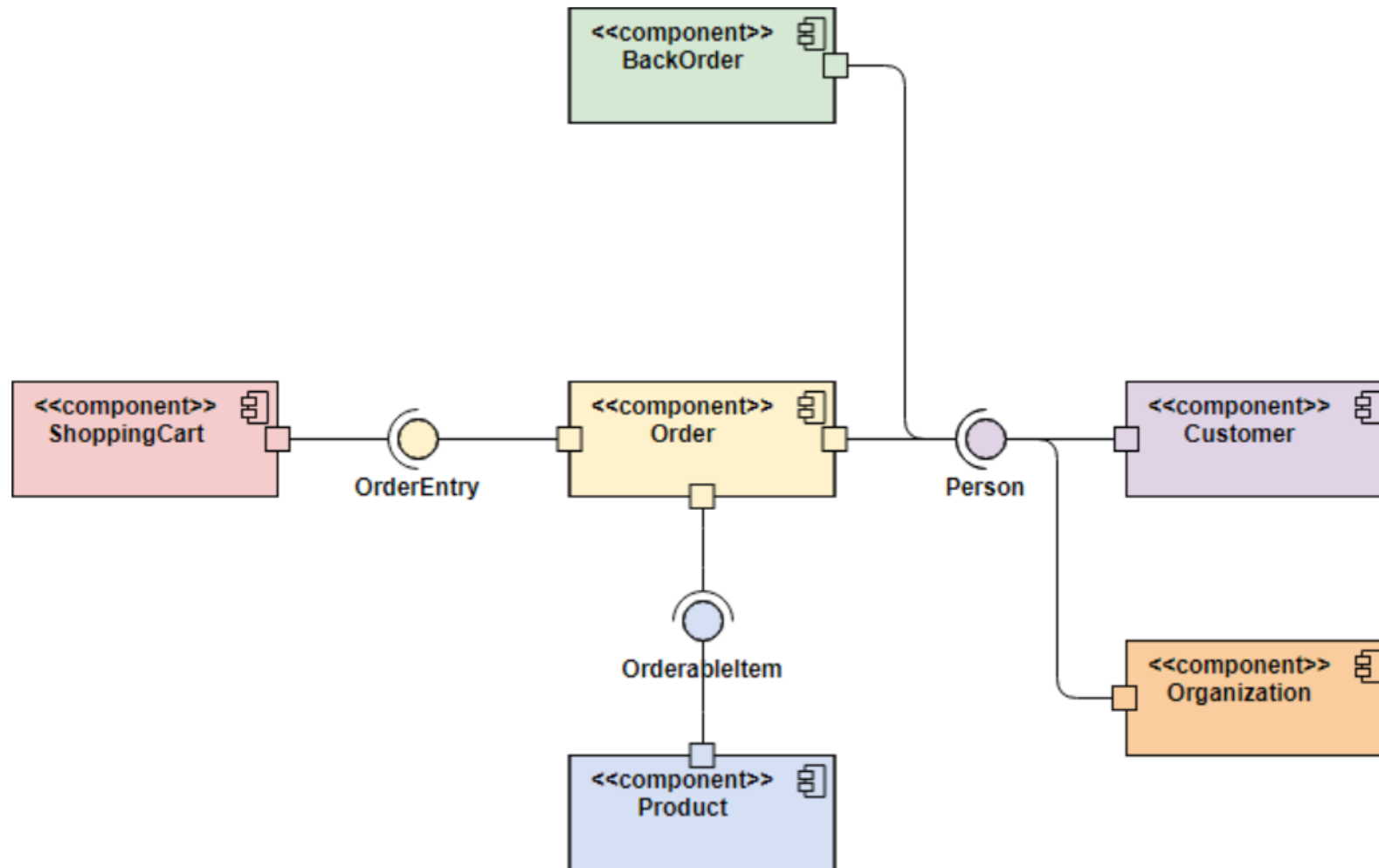
Components

- Distant cousin to the other diagrams
- Doesn't represent functionality
- Sometimes represents static implementation of the system
- Used to describe physical entities utilised by the system
- Identifies interfaces between subsystems

Component Notation



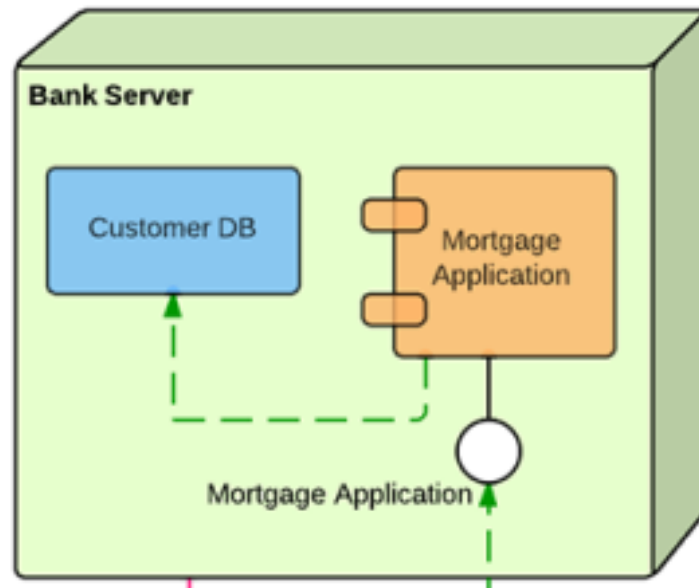
Component Diagram



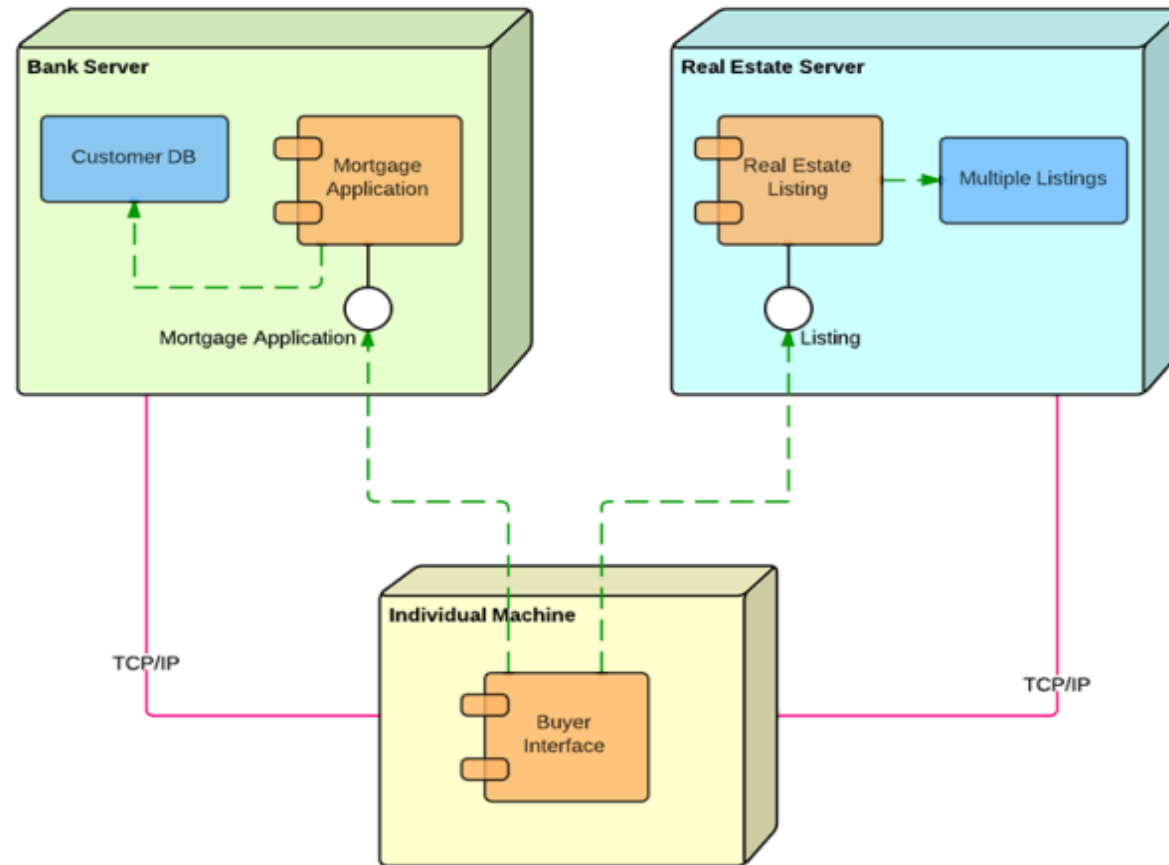
Deployment

- Provides a representation of the system as the provisioning or deployment of artefacts to one another
- Indicates what components depend upon or utilise others
- Provides a representation of included packages
- Two levels of granularity
 - Specification and Type
- Indicate Topology

Notation



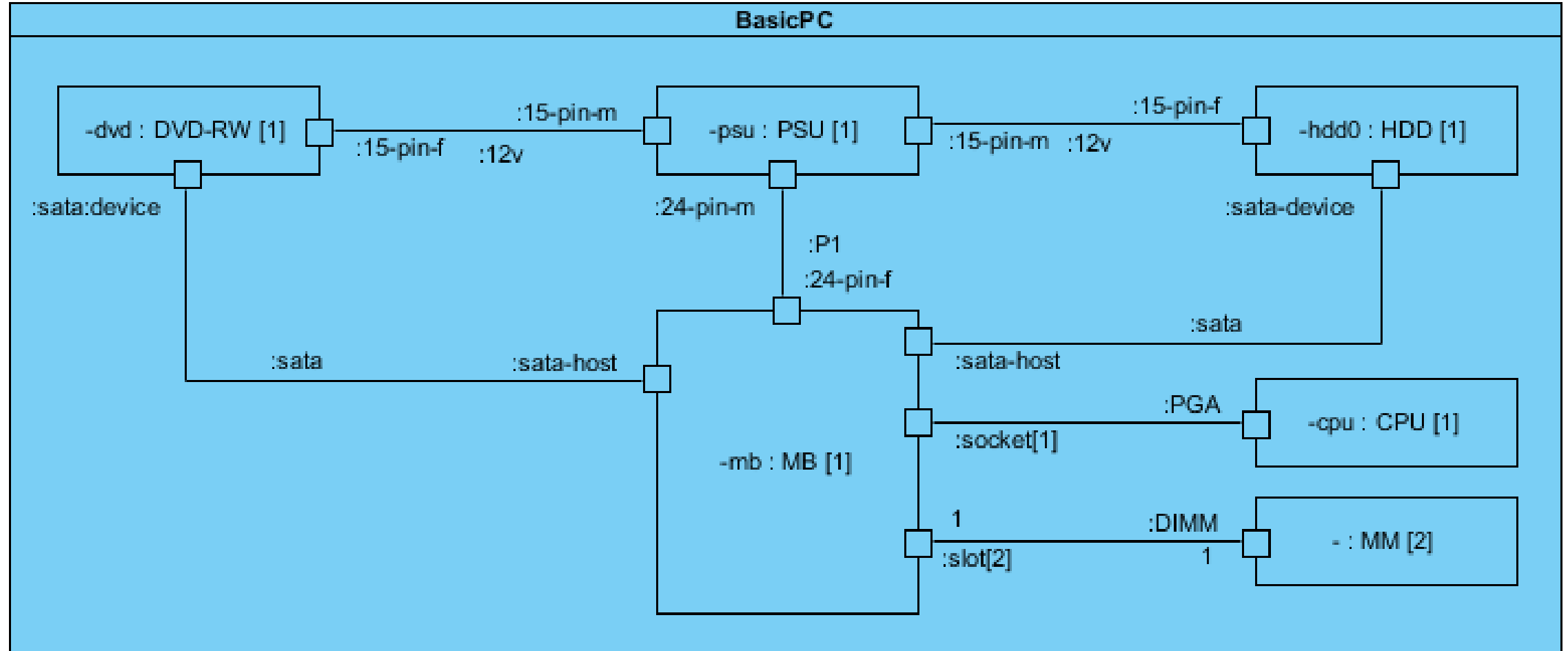
Deployment Diagram



Composites

- Static object composition
- Multiplicity of the created object
- Identifies relationships contained within the class elucidating any nesting that may occur

Composite Structural Diagram



This seems excessive

- Why do I need all of this documentation ?
 - You don't
- So which ones do I need ?
 - This is dependent entirely upon yourself and the system
 - Provide an accurate representation

Does this fit with OOAD and the AGILE Approach ?

- Almost entirely
- It follows the compartmentalisation of the system
- It allows for identification of key modules in the system

Current Trend

- UML is a staple of industry
 - Yes it is still very much utilised
- However is considered a bit excessive in some cases
- Is still used in larger companies specifically software oriented

Specific to You

- UML is useful but may sometimes be too granular
- Simple diagrammatic representation can be useful
- Not necessarily formally structured

What other Options do we have ?

- Design Absence
- Scribbles / Informal UML

Tasks

- Provide a representative structural overview of a commercial fuel pump system. For this exercise, you should represent said system utilising all of the diagrams discussed.
- Attempt to incorporate the interaction with external entities as far as possible.
- Do you feel this is a completely accurate representation of the system ?

One more thing

- This Is only half the picture
 - Behavioural diagrams are also important – this is where the stick men come in