1. *Software Systems Complexity, OO Paradigm, UML*

Software Systems Complexity

Inherent Arbitrary Complexity

- Problem Domain Complexity
  - Expressing the Requirements
  - Changing Requirements
  - System Evolution - a necessity
- Managing the Development Process
- Possible Flexibility
- Characterizing the Behavior of Discrete Systems

Software Systems Complexity

Managing the Complexity

Divide & Rule

Decomposition

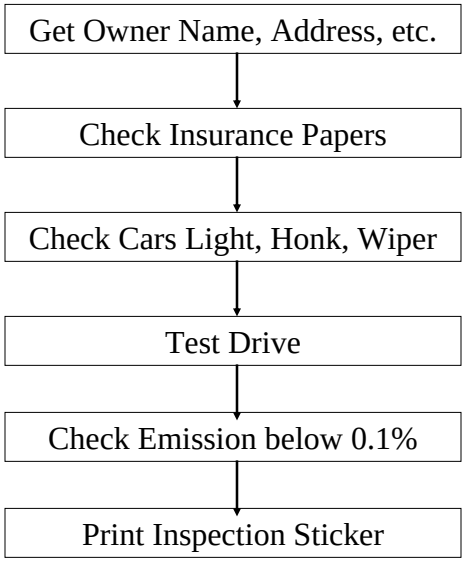
- Algorithmic Decomposition
- Object-Oriented Decomposition

Example Case Study

Example

Department of Transportation want you to build a software program that will inspect a Car based on some criteria (like checking the lights, emission control [less than 0.1% CO level], etc.) and print an inspection sticker.

Algorithmic Decomposition



Implementation - Procedural Language

```
getOwnerInfo();
checkInsurancePapers();
struct Car* carPtr = readCarInfo();
InspectCar(carPtr);
printCarInspectionSticker(carPtr);
```

InspectCar(carPtr) performs the following:

```
    checkLightsHonk(carPtr);
    testDrive(carPtr);
    checkCarEmission(carPtr);  /*Check below 0.1%*/
```

Implementation of Algorithm using Objects

```
Vehicle aVehicle = getVehicle();

Owner owner = aVehicle.getOwner();

owner.getOwnerInfo();

owner.getInsurance().getInsuranceInfo();

aVehicle.InspectVehicle();
```

InspectVehicle performs the following:

```
checkLightsHonk();
checkEmission();        // Whatever is the level
printInspectionSticker();
```

Whats the point ?!

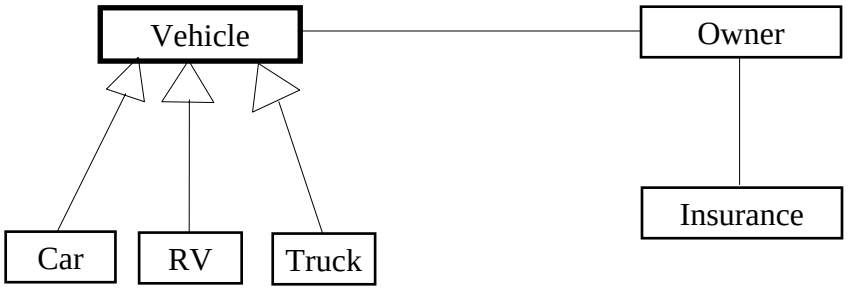
- Inspect not just Cars - inspect Trucks and RVs

Procedural Approach - Ripple Effect

```
switch (VehicleType)      // Procedural approach
{
  case CAR :
     readCarInfo, check Emission < 0.1%, etc.
     break;
  case RV:
     readRVInfo, check Emission < 0.05%,
     check Sewer System
     break;
}
```

OO Approach

- No Change to your existing code (almost)
- Merely extend the Object Model



Object-Oriented Paradigm

Collection of Discrete Objects - Data & Behavior

OO Paradigm

- Abstraction
- Encapsulation
- Classes & Objects
- Hierarchy
  - Inheritance hierarchy (“is-a”)
  - Part of hierarchy (“has a”)
- Polymorphism

Abstraction

“A simplified description ... of a system that emphasizes some of the system’s details ... while suppressing others”

“An abstraction denotes the essential characteristics of an object that distinguish it from all other kind of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer”

Encapsulation

Information hiding

Interface - Implementation

Behavior & Data

"Encapsulation is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation"

Hierarchy

"Hierarchy is a ranking of abstractions"

Inheritance : expresses "is-a" or "Kind-of" relationship

- Extensibility & Reusability

Part-of: expresses that an object is an aggregate of other objects

Polymorphism

Hiding alternative procedures behind a Common Interface

Send a Message to an object - Polymorphism guarantees that the correct/proper implementation is invoked.

OO Programming Languages

Traditional Structured	Object-based	Class-based	Pure Object-Oriented	Hybrid Object-Oriented
FORTRAN C Pascal COBOL	Ada	CLU	Java Smalltalk Eiffel Simula Trellis	C++ Objective-C CLOS Object- Pascal Object-COBOL

OOPLs

- Pure OOPLs
  - Supports the Paradigm
  - Enforces the Paradigm
- Hybrid OOPLs
  - Supports the Paradigm
  - Not Strictly Enforced

Unified Modeling Language (UML)

- Notations -
  Express & Communicate Classes, Relationships, etc.
- OMG - Booch, Rumbaugh, Jacobson, et. al.
- Notations Subset Introduced When Used

OO Features in C++

- User-defined data types - Classes
- Inheritance
- Virtual Functions (Polymorphism)
- Function Overloading
- Operator Overloading
- Templates

Other Features in C++

- Strong Type Checking
- Flexible Declarations
- Scope & Scope resolution ::
- const declaration
- void & void pointers
- Enhanced I/O
- Reference
- new & delete

Example C++ Program

The C++ standard library provides objects that facilitate easy input/output.

```
#include <iostream.h> // Include the standard iostream library header file
void main()
{
    cout << "Hello World" << endl;          // print Hello World on a line

    cout << "Please enter an integer:";     // print a prompt message

    int intval;                             // declare a variable of type int

    cin >> intval;                          // input an integer value from the keyboard (stdin)

    cout << "You entered " << intval << endl; // Output a message with the value provided
}
```

Output:

```
Hello World
Please enter an integer:6
You entered 6
```

Lab Work: Details provided on-line.