ENSF 480
Principles of Software Design

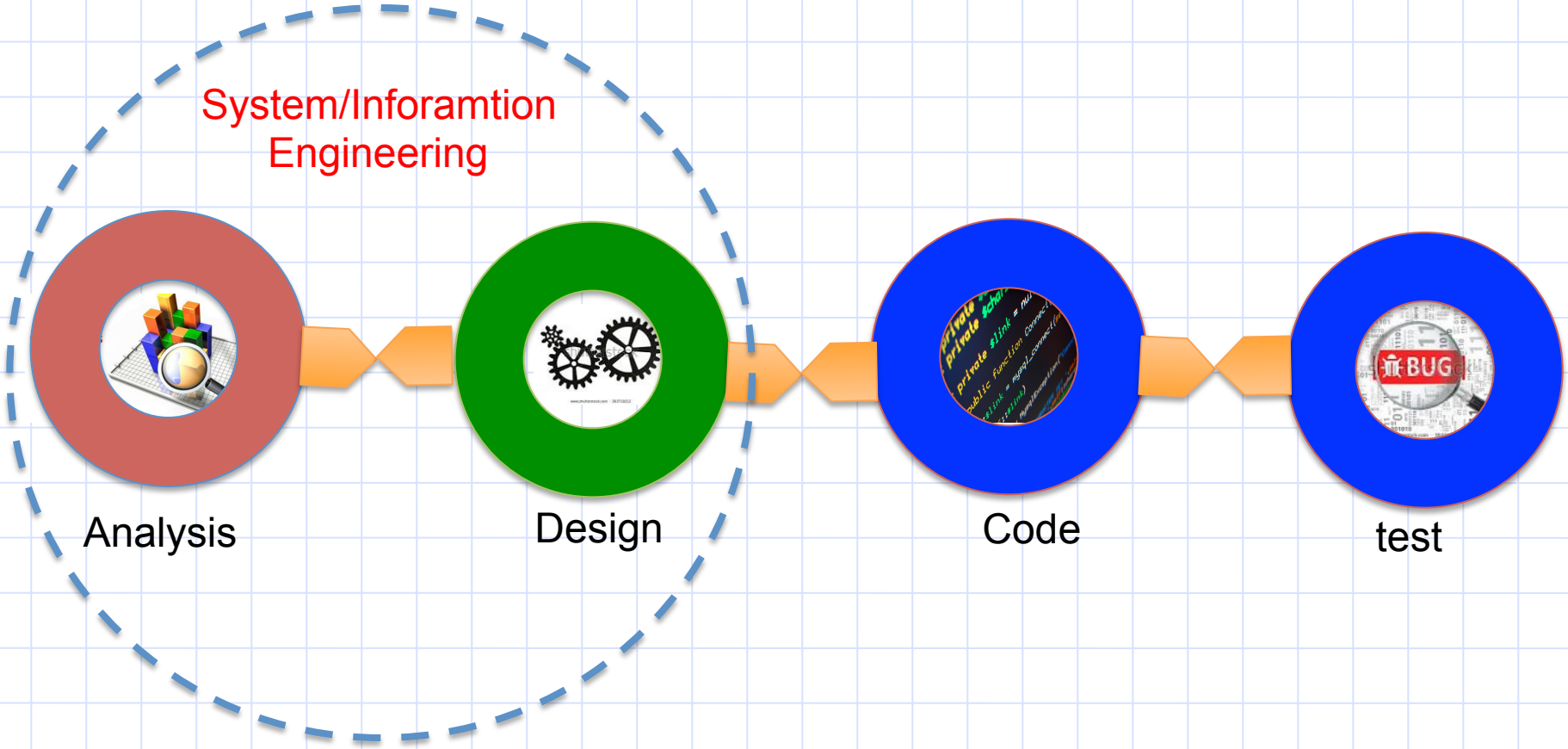
A Summary on the History

- Early 1960's:
  - System Development Life Cycle (SDLC), the oldest methodology framework for building information systems.
- 1970s:
  - Structured System Analysis and Design (SSADM)
- 1990s:
  - Object-Oriented Analysis and Design (OOAD)
- More Recent Concepts
  - Agent-Oriented Systems (AOS)
  - Service-Oriented Systems (SOS)

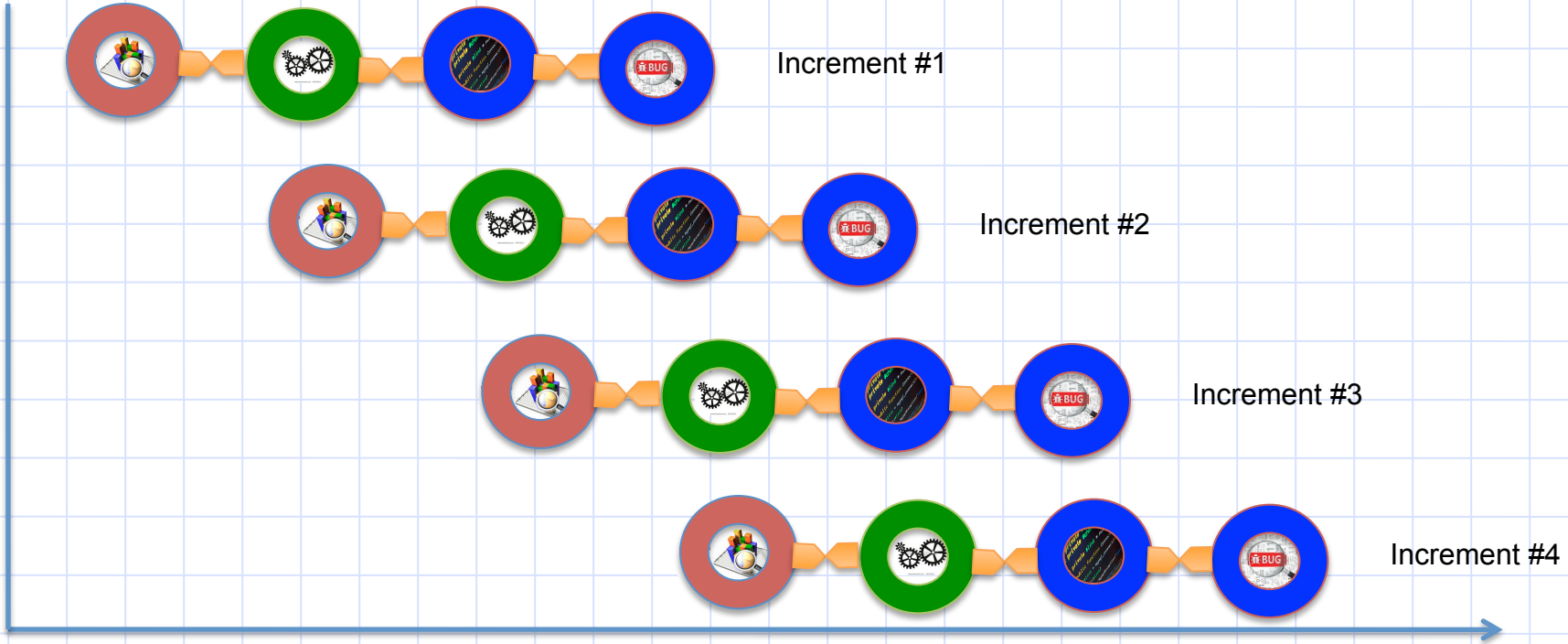
A Brief Discussion on Software Development Process

A Simple Linear Model

- A Software Development Process Model defines the order of the phases of software development activities and the software lifecycle.

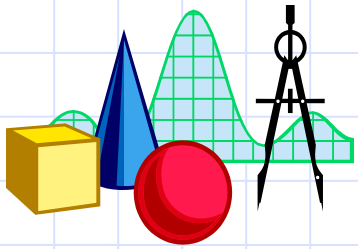


The Incremental Model

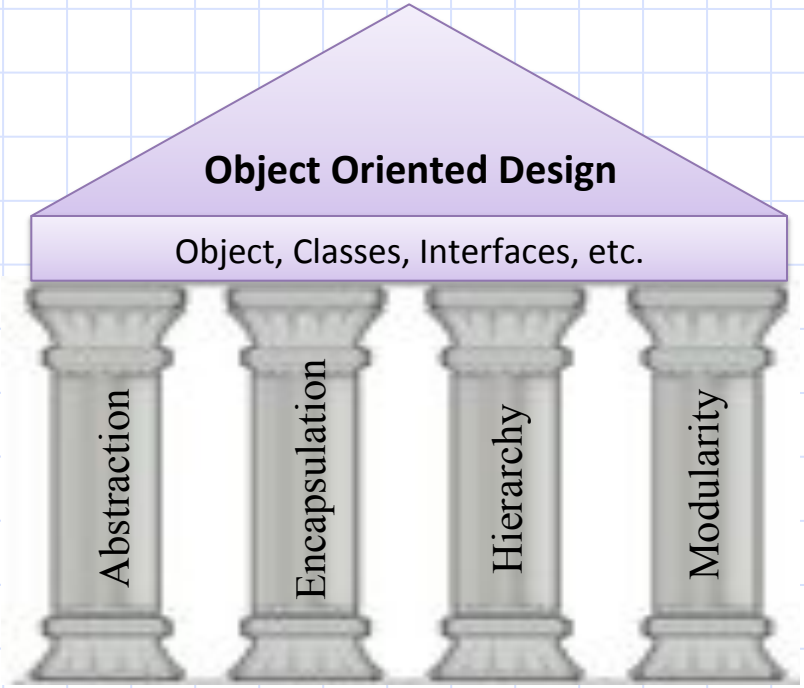


- There are many more models that we will briefly discuss later in this course and in more details in SENG 511:
  - Iterative Models
  - Spiral Model
  - V-Shape Model
  - Agile Model
  - Etc.

Review of Object Technology



Major Pillars of Object-Oriented Methodology?



- There are four major elements of the object models:
- Abstraction
- Encapsulation
- Hierarchy
- Modularity

What is Abstraction?

Abstraction

- Abstraction is a technique of dealing with complex system. We make a simplified model of a complex system.
  - By abstraction, we ignore the inessential details.
  - An abstraction focuses on the outside view of an object.
    - Properties
    - Outside view of behavior
- "Deciding upon the right set of abstractions for a given domain is the central problem in object-oriented analysis and design."
- In context of Object Oriented Programming, abstraction is represented by a "class" definition .

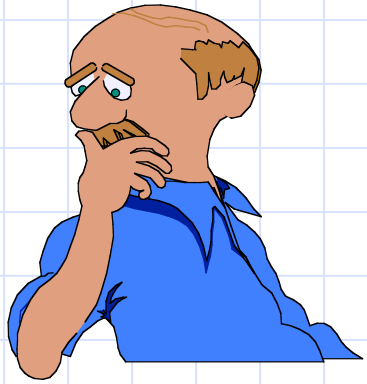
Abstraction Example in C++

Outside view Abstraction

Outside view Abstraction in C++

```
class Calculator
{
   private:
      char * operands;
      double result;
      bool isOn;
      …
   public:
      Calculator();
      double add();
      double subtract();
      …
};
```

Class Discussions



- Assume, as a junior software engineer, and as part of team that develops an software application for a library, you have been assigned to design a class called "Book".
  - Please get into groups of 2 or 3, discuss the issue and come up with some solution?

Discussions cont'd: One possible solution in C++

```
class Book{
    private:
                char* title;
                char** authors;
                char* publisher;
                int numberOfAuthors;
                int yearPublished;
                char* bookState;
      public:
            Book(….);
            ~Book();
            Book (const Book& src);
            Book& operator = (const Book& rhs);
            void bookInfo();
            // a set of getters
            …
            // a set of setters
};
```

Other Concerns

- Is this class subject to any hierarchical structure or classes?
  - Inheritance?
  - Aggregation?
  - Composition?
- Lets assume we would like to have "Book" as a base-class for a few other classes:
    - Textbooks
    - Novels
    - Etc.

Discussions cont'd: One possible solution in C++

```
class Book{
    private:
                char* title;
                char** authors;
                char* publisher;
                int numberOfAuthors;
                int yearPublished;
                char* bookState;
       public:
            Book(….);
            virtual ~Book();
            Book (const Book& src);
            Book& operator = (const Book& rhs);
           virtual void bookInfo();
            // a set of getters
            …
            // a set of setters
};
```

Discussions cont'd: Other possible solution in C++

```
class Book{
    private:
                string title;
                vector<string> authors;
                string publisher;
                int numberOfAuthors;
                int yearPublished;
                string bookState;
       public:
            Book(….);
           virtual bookInfo();
            // a set of getters
            …
            // a set of setters
};
```

Discussions cont'd: A Java Solution

```
class Book{
    private String title;
    private ArrayList<String> authors;
    private String publisher;
    Integer numberOfAuthors;
    Integer yearPublished;
    String bookState;
    public Book(….) {}
    abstract bookInfo();
    // a set of getters
            …
    // a set of setters
}
```

What is Encapsulation?

Encapsulation

- Encapsulation is achieved through information hiding:
  - It refers to inside view of the object
  - Although the interface of the methods are public, but the implementation detail of the objects and the current values of its data are hidden.
  - We can re-implement anything inside the object's capsule without affecting other objects that interact with it.

Example

– Inside view – the details that are hidden

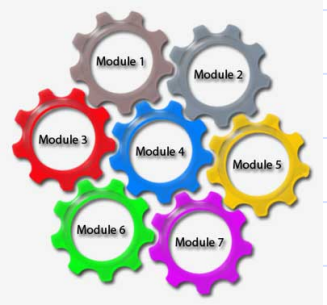
Inside view - Encapsulation

Inside view – Encapsulation in C++

```
Calculator::Calculator() {
  // some code
}

double Calculator::add() {
  // some code
}

double Calculator::subtract() {
  // some code

}
```

What is Modularity?

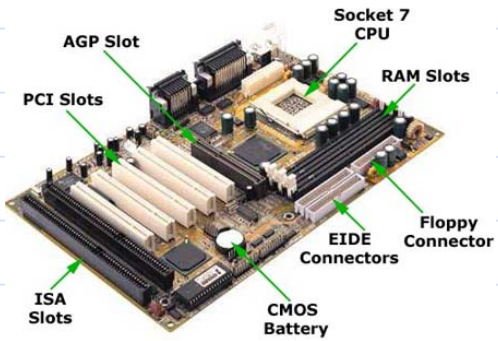
Modularity



- The property of a system decomposed into a set of cohesive and ***loosely*** coupled modules.
- Modules are *physical* containers in which classes and objects (the logical design) are placed.
- A module has an *interface* and a *body* (implementation).
  - Changing the body requires recompiling just that module.
  - Changing the interface requires recompiling the module, plus all other modules that depend on the interface.

Examples

- Hardware versus software.




```
class Employee {
   private:
      string name;
      string address;
      string birthday;
   public:
      string getName() const;
      void setNam(string name);
      …
      …
};
```

```
class Company {
   private:
      string name;
      string address;
      string dateStablished
   public:
      string getName() const;
      void setNam(string name);
      …
      …
};
```

```
class project {
   private:
      string title;
      string dateStablished
   public:
      string getName() const;
      void setName(string name);
      …
      …
};
```

Can we make it more modular?

Examples in C++

Discuss the advantages and disadvantages of more decomposition

```
class Company {
   private:
      string name;
      Address address;
      Date dateStablished
   public:
      string getName() const;
      void setNam(string name);
      …
      …
};
```

```
class Employee{
   private:
      Name name;
      Address address;
      Date birthday
   public:
      …

};
```

```
class project {
   private:
      string title;
      Address address;
      Date dateStablished
   public:
      string getName() const;
      void setName(string name);
      …
      …
};
```

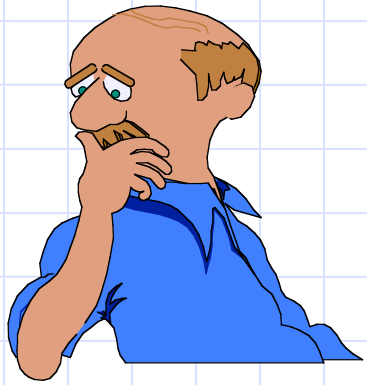
More decomposition

```
class Address{
   private:
      string street;
      string city;
      string postalCode
      string country
   public:
      …
};
```

```
class Name{
   private:
      string firstname;
      string midname;
      string lastname
   public:

      …
};
```

```
class Date{
   private:
      string day;
      string month;
      string year;
   public:

      …
};
```

Class Discussions



- Considering the class Book in one of the previous slides, discuss the possible options to improve its modularity.

Discussions cont'd: One Possible Suggestion

```
class Book{
    private:
                string title;
                vector<Author> authors;
                Publisher publisher;
                int numberOfAuthors;
                Date published;
                BookState bookState;
      public:
            Book(….);
           virtual bookInfo();
            // a set of getters
            …
            // a set of setters
};
```

```
class Date{
    …
};

class Author{
    …
};
class Publisher {
     …
};
class BookState {
     private:
         bool isAvailable;
         bool isDamaged;
         bool IsReturned
};
```

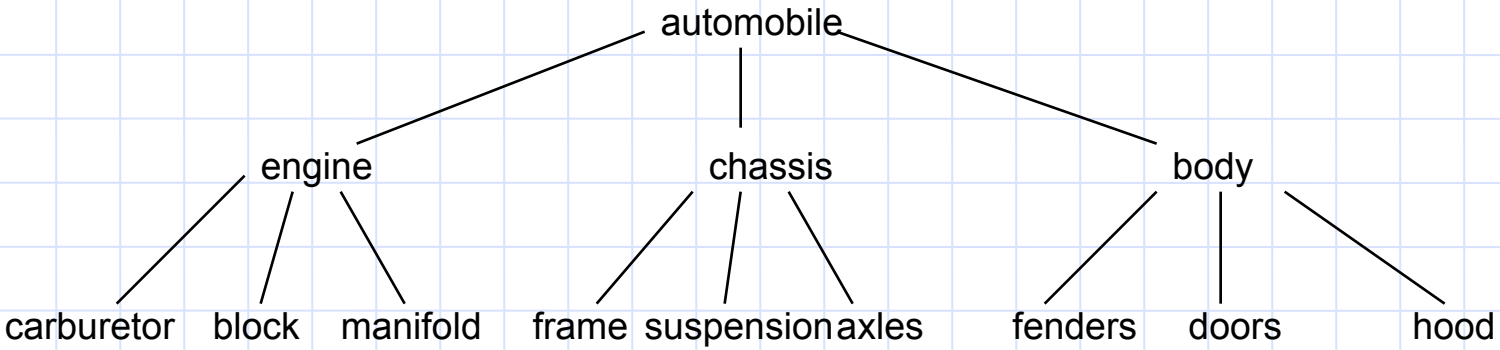
What is Hierarchy?

Hierarchy

- Object-oriented systems support two types of hierarchy:
  - Whole-Part hierarchy: represents containment relationship.
  - Generalization-specialization hierarchy: represents parent/child relationship.

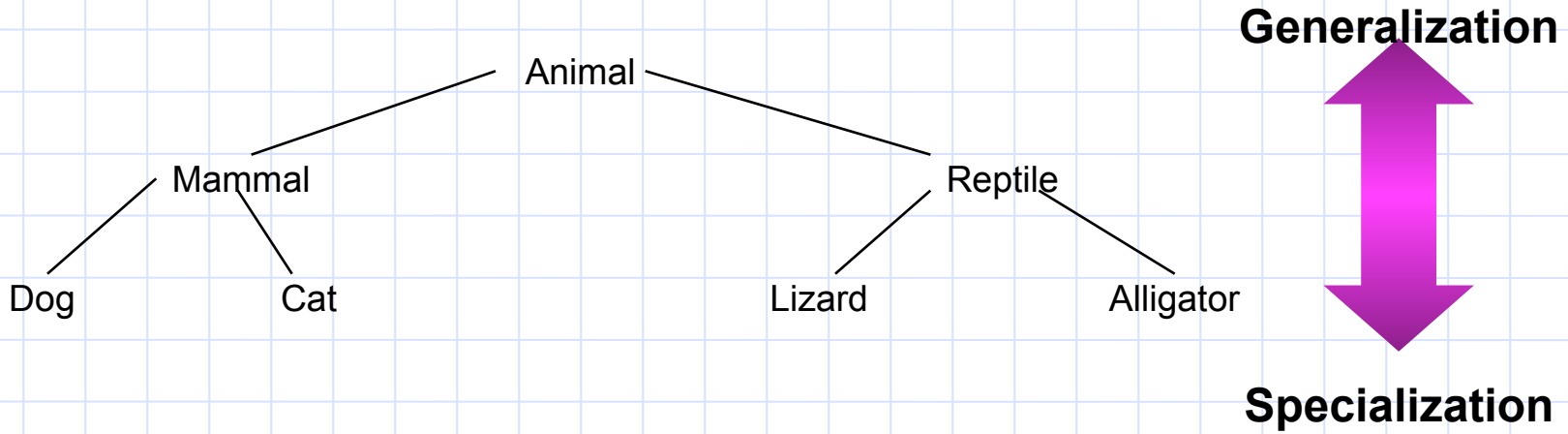
What is Hierarchy?

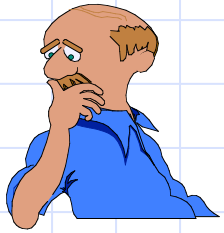
- Systems may have one of the two type of consist hierarchies:
  - "has a" hierarchy
    - Often called the *object structure.*



Hierarchy (continued)

- "Is a" hierarchy
  - Objects are specialization of more general kinds of objects.





Class Discussions

- Fred is a Dodge dealer. His dealership maintains and sells a vehicle inventory typically numbering over 100 at any given time. It also services cars of all makes. while it stocks a variety of parts for Dodge models, it purchases parts for vehicles of other brands on an as needed basis. What "kind of" and “Part of” hierarchies might be useful in organising Fred's dealership?