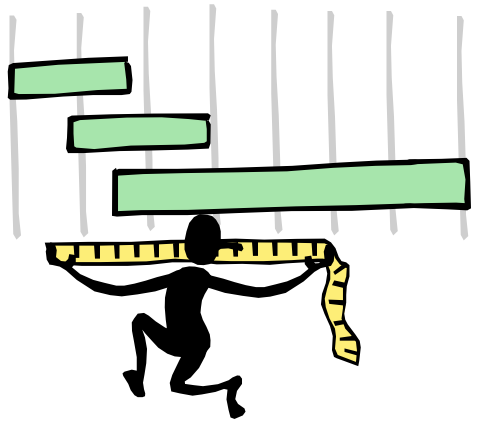
Object Oriented Analysis and Design

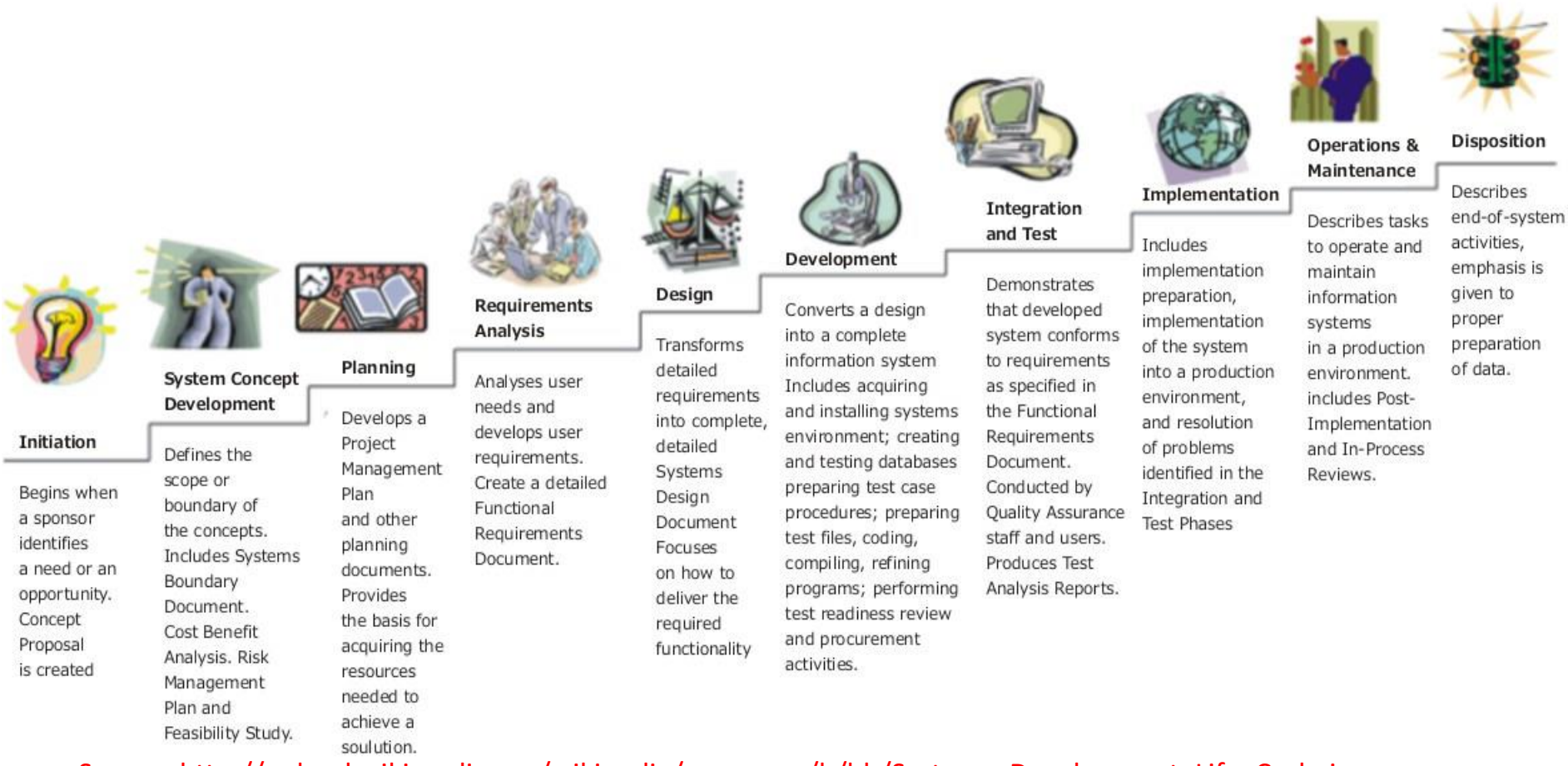
System/Software Life Cycle (SLC)

- Life cycle is the series of steps that software undergoes from concept exploration through retirement

- Intended to develop information systems in a very **deliberate**, **structured** and **methodical** way, reiterating each stage of the life cycle.



Systems Development Life Cycle (SDLC)
Life-Cycle Phases



Source: http://upload.wikimedia.org/wikipedia/commons/b/bb/Systems_Development_Life_Cycle.jpg

Importance of Lifecycle Models

- Provide guidance for project management
  - what major tasks should be tackled next? milestones!
  - what kind of progress has been made?
- The necessity of lifecycle models
  - characteristics of software development has changed
    - early days: programmers were the primary users
    - modest designs; potential of software unknown
  - more complex systems attempted
    - more features, more sophistication → greater complexity, more chances for error
    - heterogeneous users

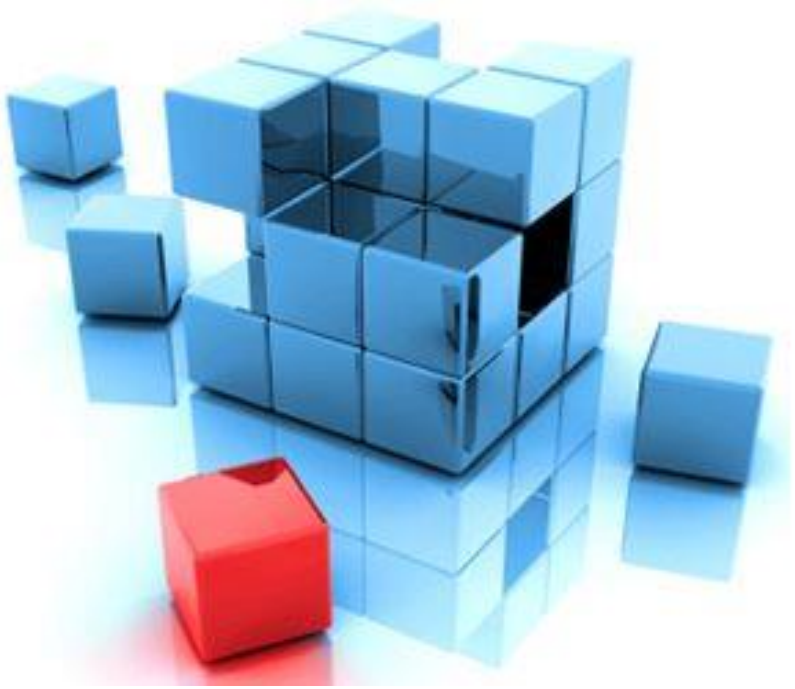
Object Oriented Approach

- Software development based on modeling objects from the real world and then use the models to build a language independent design organized around those objects.
- Promote better understanding of requirements, cleaner designs, and more maintainable systems.
- Use language-independent graphical notation for analyzing problem requirements, design a solution to the problem and then implement the solution.
- Same concept and notation throughout the software development process.
- Object oriented concepts throughout the software life cycle, from analysis through design to implementation.

Cont.

- Coding - last stage in the process of development.
- A good design technique defers implementation details until later stages of design to preserve flexibility.
- Mistakes in the front of development process have a large impact on the ultimate product and on the time needed to finish.
- OOT is a way of thinking abstractly about a problem using real world concepts, rather than a computer concepts.
- Graphical notations helps developer to visualize a problem without prematurely resorting to implementation

Why Modeling..?



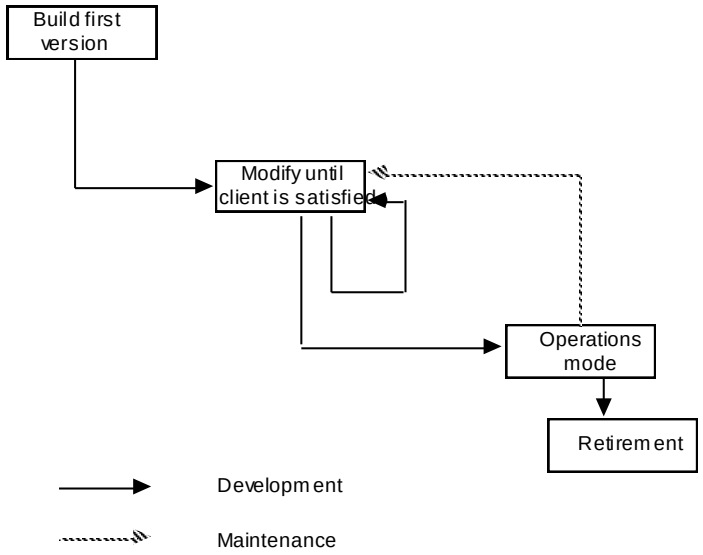
Modeling

- A model is a simplification of reality
- A hypothetical description of a complex entity or process
- A good model includes those elements that have broad effect and omits minor elements
- A model of a system is not the system!
- Modeling is used in many disciplines – architecture, aircraft building, …

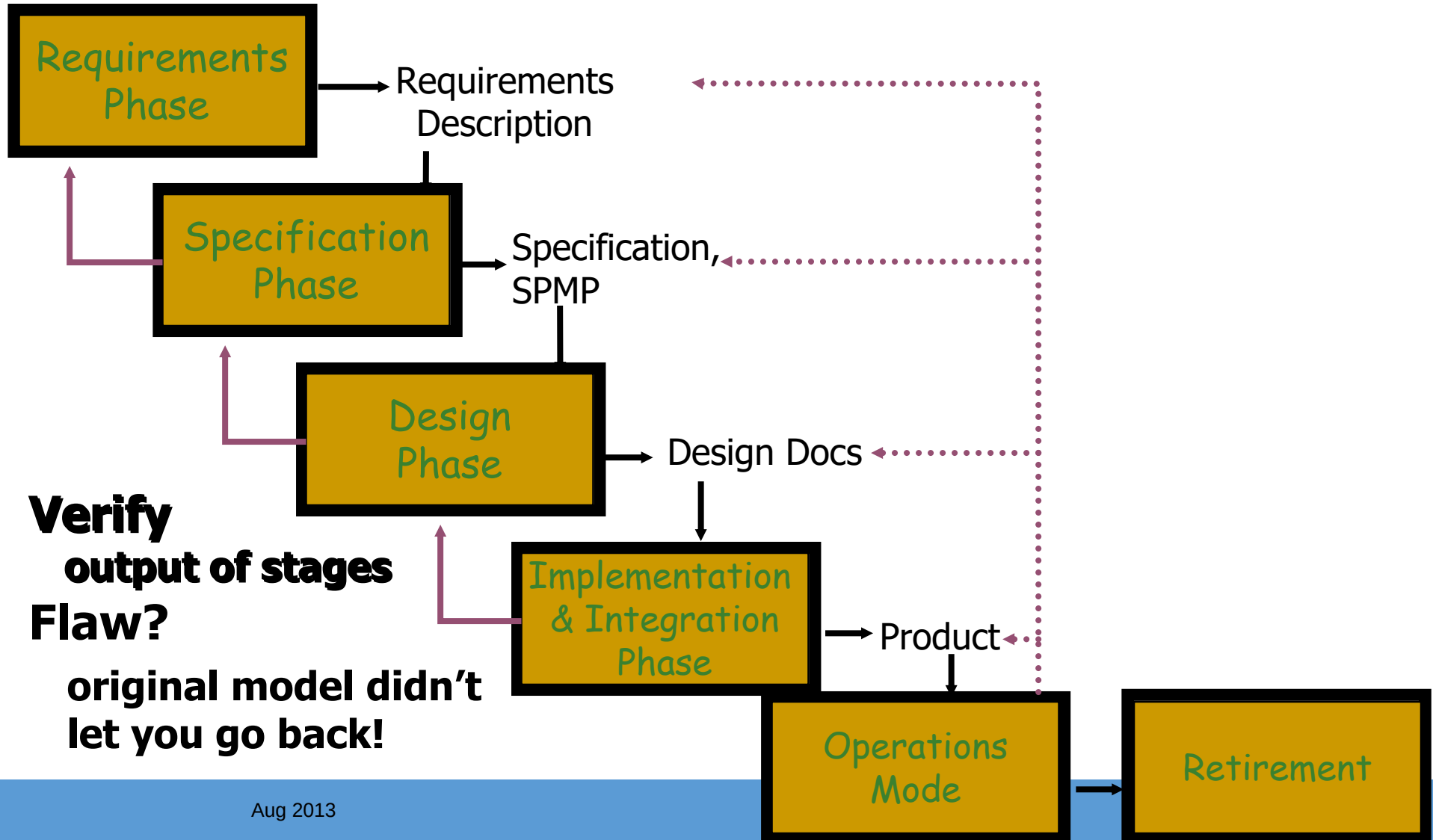
Why Modeling..?

- ✓ Models help us to **visualize** a system as it is or as we want it to be.
- ✓ Models permit us to **specify the structure or behavior** of a system.
- ✓ Models give us a **template** that guides us in constructing a system.
- ✓ Models document the **decisions** we have made.

Build and Fix Model



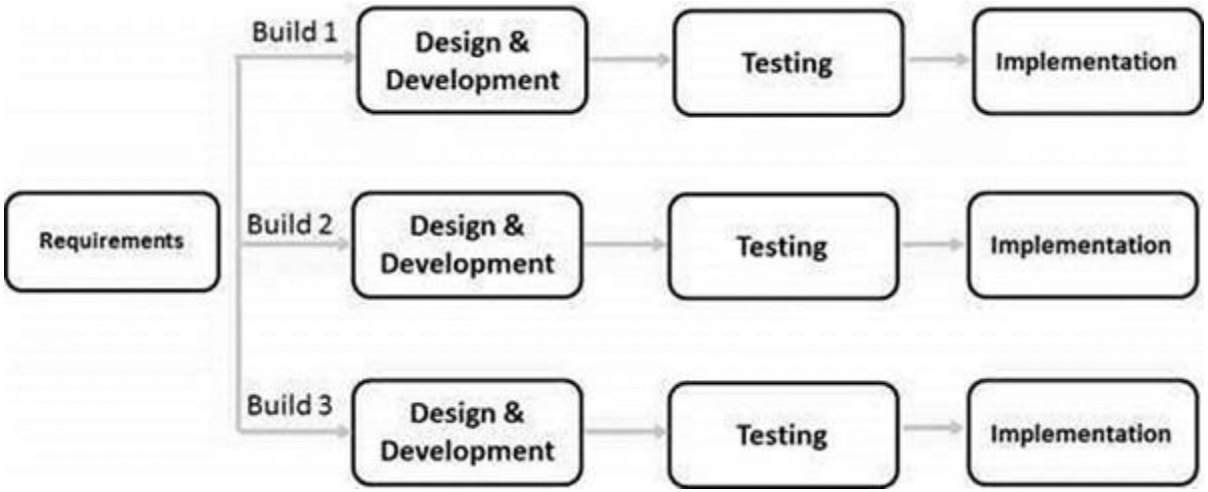
Waterfall Model



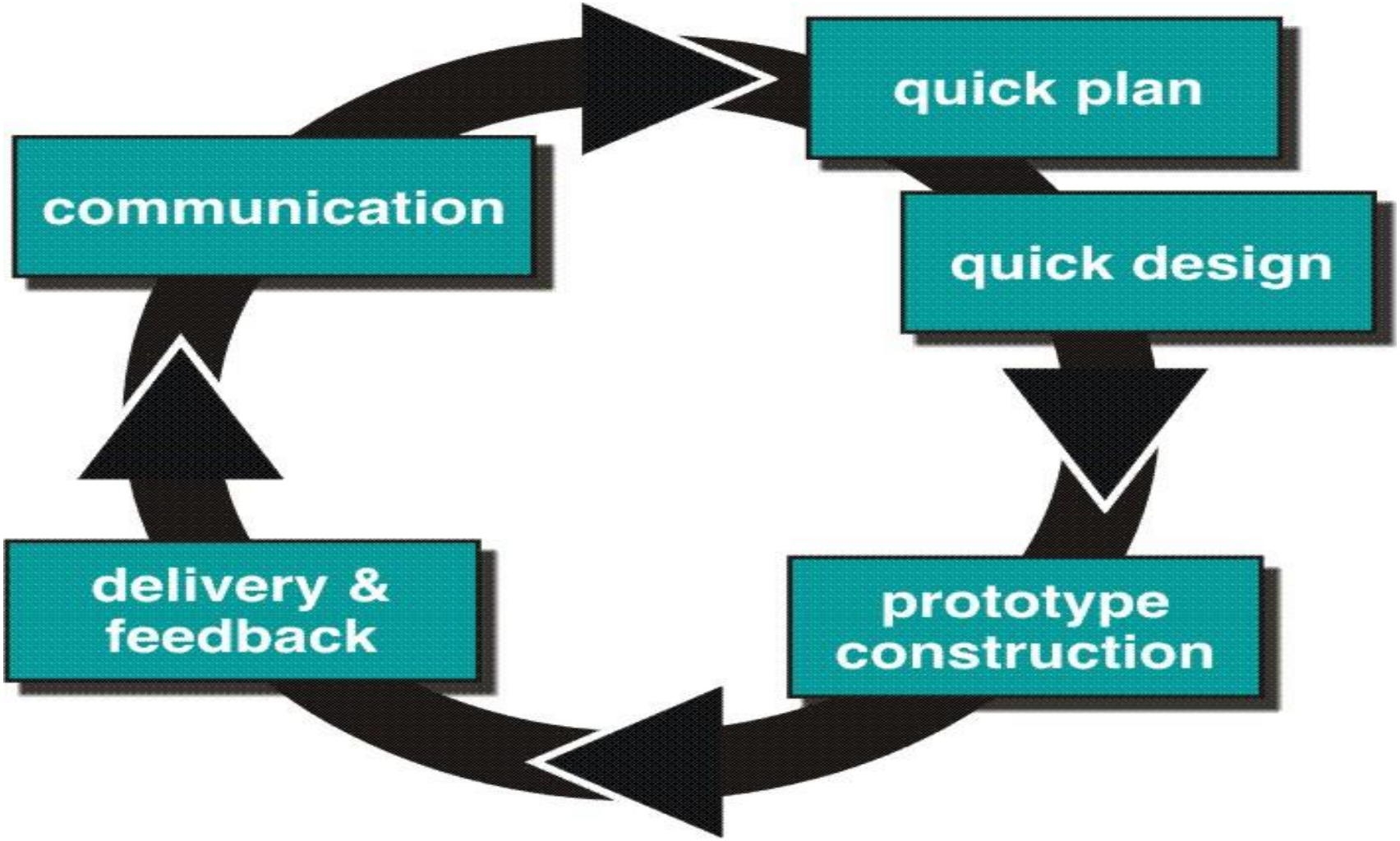
Verify
output of stages

Flaw?
original model didn't let you go back!

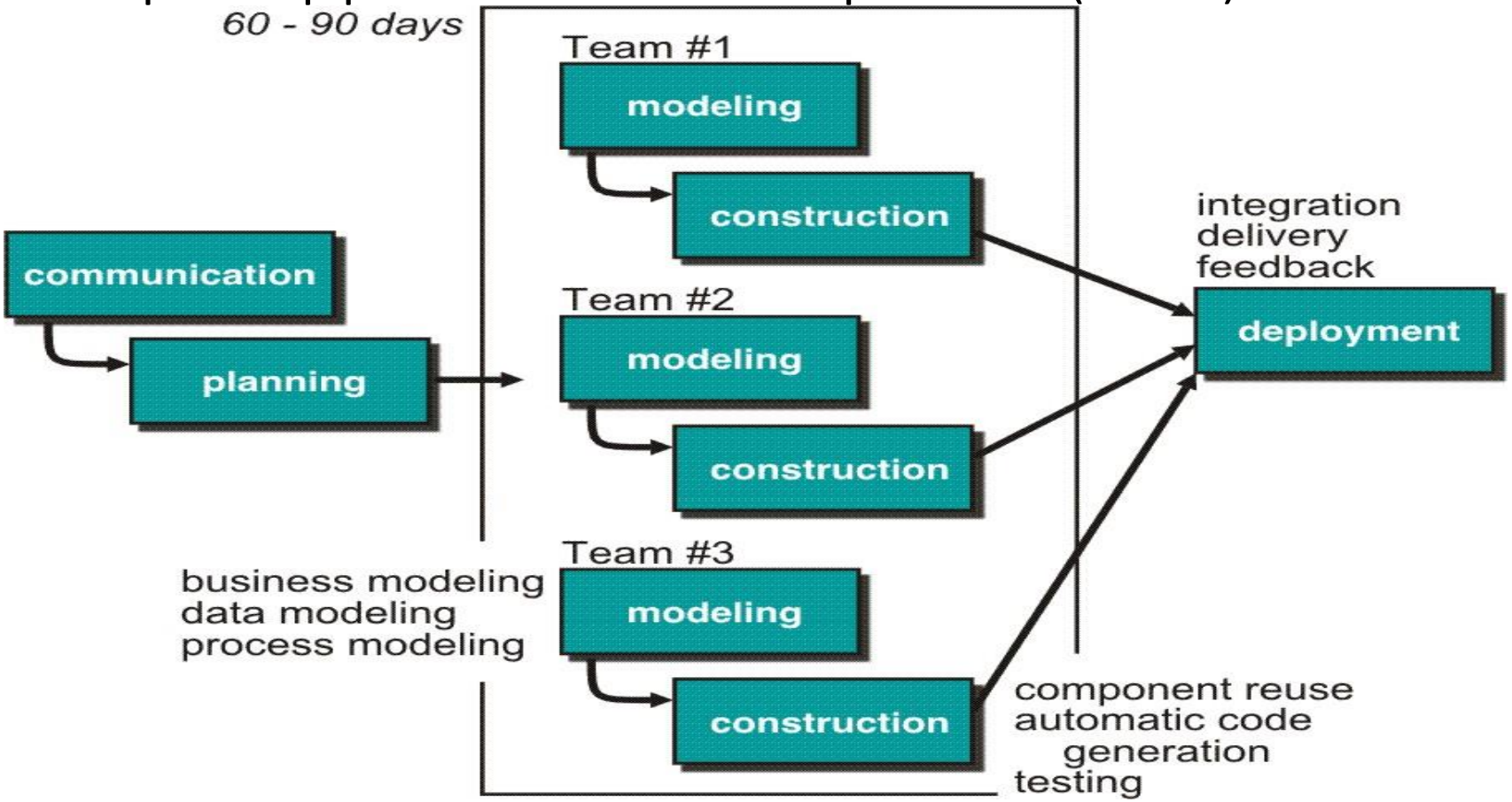
Iterative Model



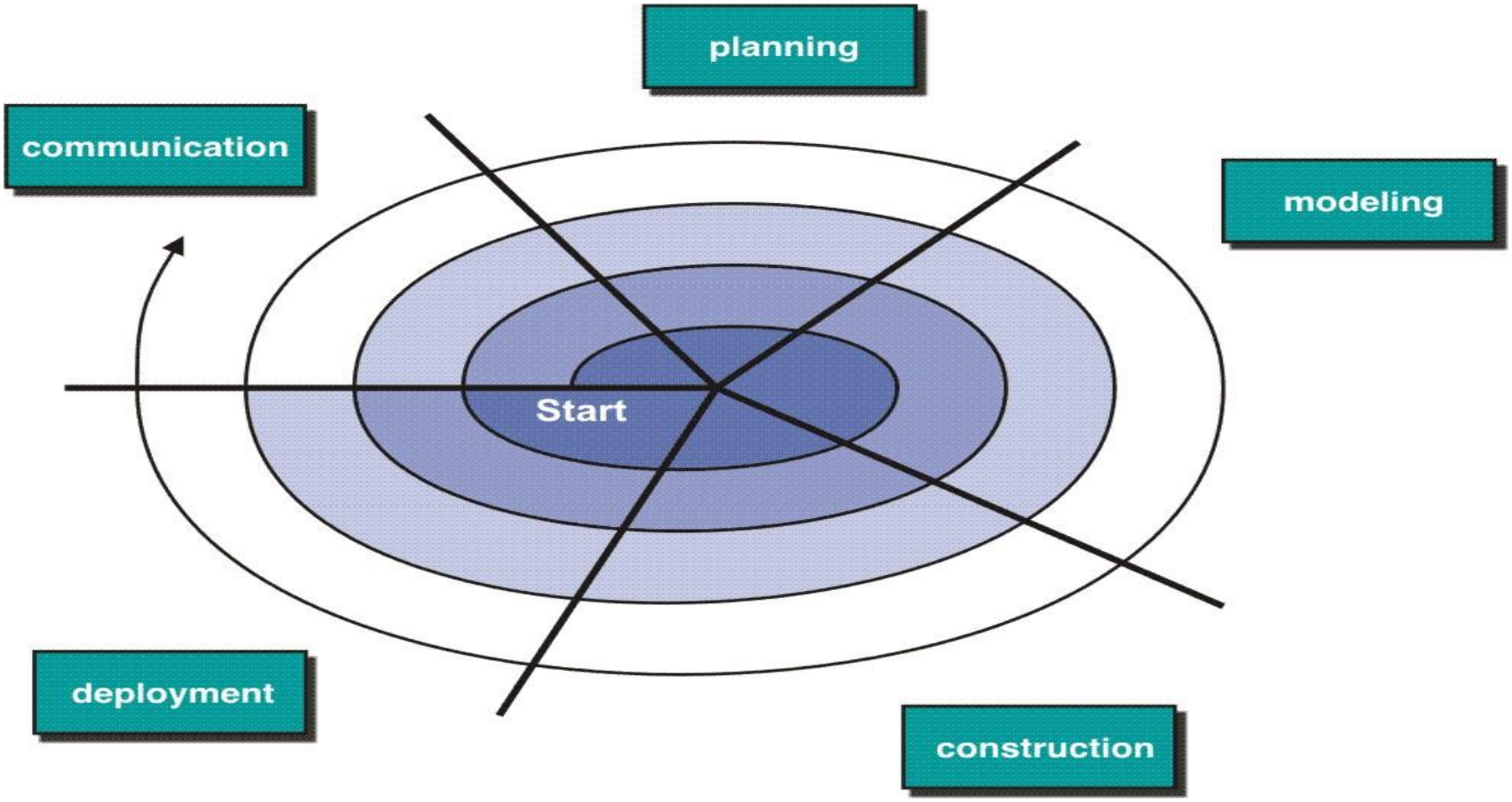
Prototype Model



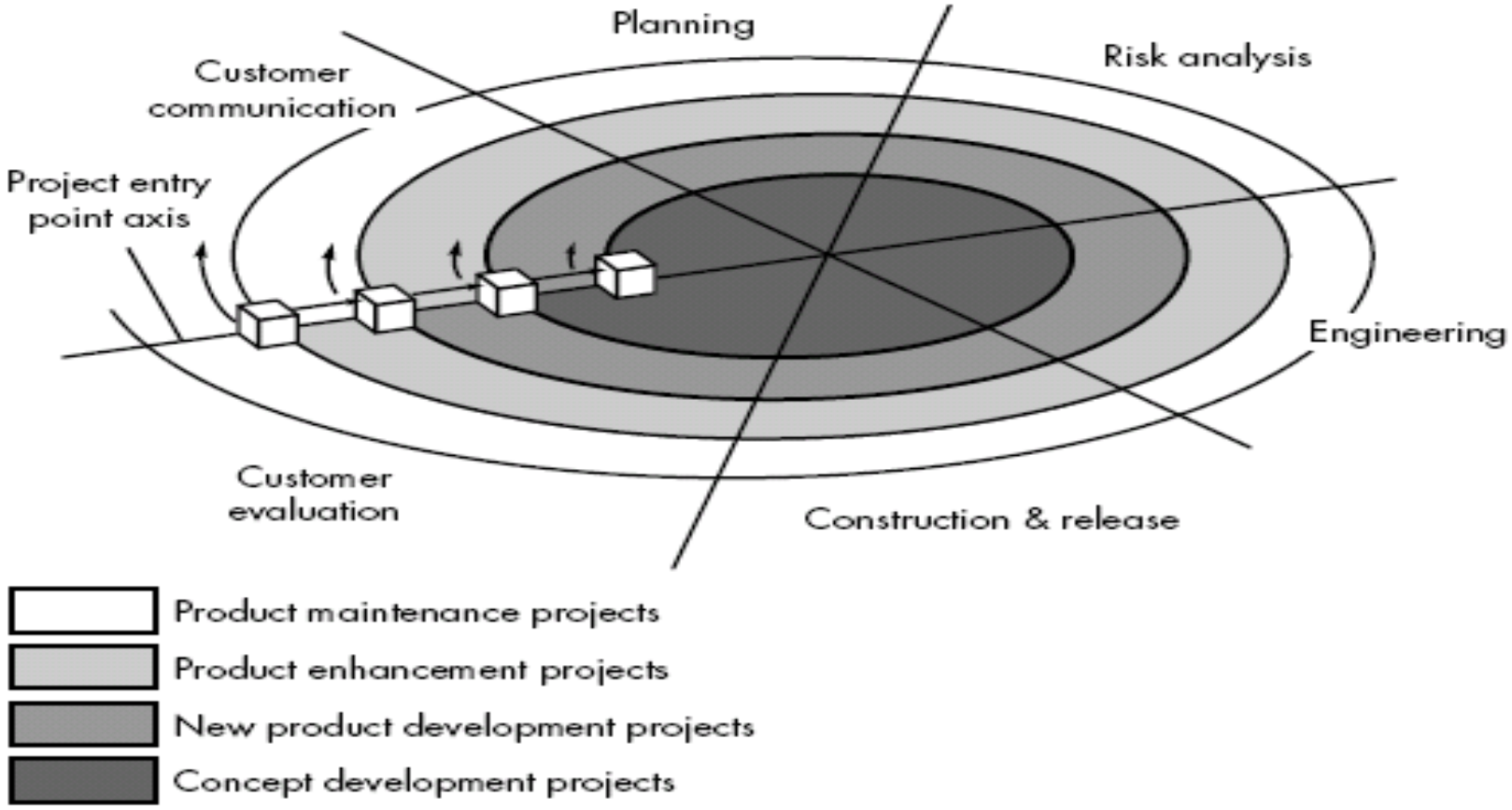
Rapid Application Development (RAD) Model



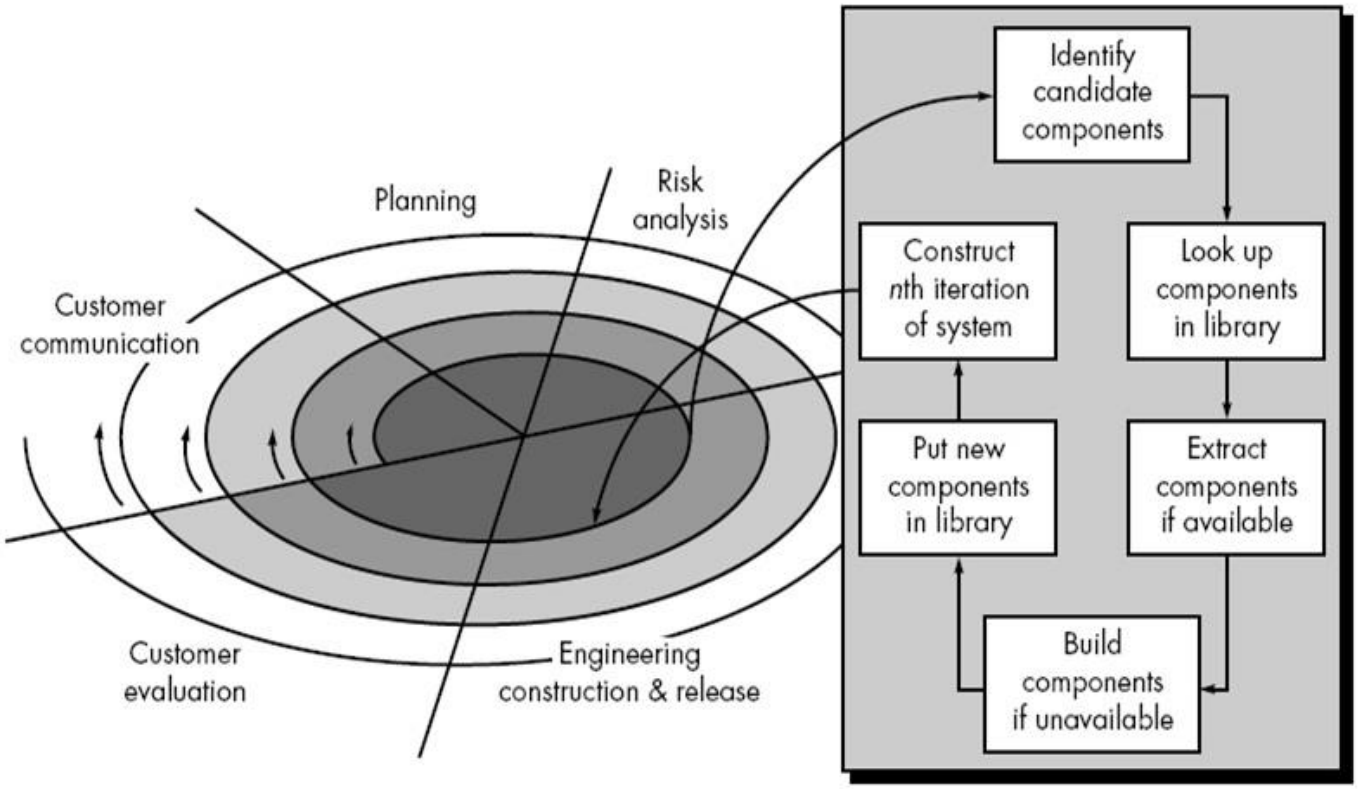
Spiral Model



Spiral Model



CBD Model


Planning
Risk analysis
Customer communication
Customer evaluation
Engineering construction & release
Identify candidate components
Construct nth iteration of system
Look up components in library
Put new components in library
Extract components if available
Build components if unavailable

INTRODUCTION

- An Overview of Object Oriented Systems Development
- Object Basics
- Object Oriented Systems Development Life Cycle

Object-oriented analysis anddesign

- Object-oriented analysis and design (OOAD) is a popular technical
  approach for
  - analyzing,
  - designing an application, system, or business
  - by applying the object oriented paradigm and
  - visual modeling throughout the development life cycles for better communication
    and product quality.

- **What is OOAD?-** Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects .

- **Analysis** — understanding, finding and describing concepts in the problem domain.

- **Design** — understanding and defining software solution/objects that represent the analysis concepts and will eventually be implemented in code.

- **OOAD -** A software development approach that emphasizes a logical solution based on objects.

- Software development is dynamic and always undergoing major change.

- System development refers to all activities that go into producing information system solution.

- System development activities consist of
  - system analysis,
  - modelling,
  - design,
  - implementation,
  - testing and maintenance.

- A software development methodology → series of processes → can lead to the development of an application.

- Practices, procedures, and rules used to develop software, totally based on system requirements

ORTHOGONAL VIEWS OF THE SOFTWARE

- Two Approaches,
  - **Traditional Approach**
  - **Objected-Oriented Approach**

- **TRADITIONAL APPROACH**

  - Collection of **programs or functions.**
  - A system that is designed for **performing certain actions**.
  - **Algorithms + Data Structures = Programs**.
  - Software Development Models (**Waterfall, Spiral, Incremental, etc..)**

Difference between Traditional and Object Oriented Approach

TRADITIONAL APPROACH	OBJECT ORIENTED SYSTEM DEVELOPMENT
Collection of procedures(functions)	Combination of data and functionality
Focuses on function and procedures, different styles and methodologies for each step of process	Focuses on object, classes, modules that can be easily replaced, modified and reused.
Moving from one phase to another phase is complex.	Moving from one phase to another phase is easier.
Increases duration of project	decreases duration of project
Increases complexity	Reduces complexity and redundancy

BENEFITS OF OBJECT ORIENTATION

- **Faster development,**
- **Reusability,**
- **Increased quality**
- modeling the **real world and provides us with the stronger equivalence of the real world‘s entities (objects)**.
- Raising the level of abstraction to the point where **application can be implemented in the same terms as they are described.**

WHY OBJECTORIENTATION

- OO Methods enables to develop **set of objects that work** together.
- It adapts to

  - **Changing requirements**
  - **Easier to maintain**
  - **More robust**
  - **Promote greater design**
  - **Code reuse**

- Others

  - **Higher level of abstraction**
  - **Seamless transition among different phases of software development.**
  - **Encouragement of good programming technique.**
  - **Promotion of reusability.**

OVERVIEW OF UNIFIED APPROACH

- The **unified approach (UA)** is a methodology for **software development.**

- **Booch, Rumbaugh, Jacobson methodologies** gives the best practices, processes and guidelines for OO oriented software development.

- Combines with the **object management groups in unified modelling language.**

- UA utilizes the **unified modeling language (UML)** which is a set of **notations and conventions** used to describe and **model an application.**

Continued…

Layered Architecture

- UA uses **layered architecture to develop applications**.
- **Creates object** that **represent elements to the user** through **interface or physically stored in database.**
- The layered approach consists of **user interface, business, access layers.**
- This approach **reduces the interdependence of the user interface, database access and business control.**
- **More robust and flexible system.**

OBJECTBASICS

Goals:

The developer should

- Define Objects and classes
- Describe objects, methods, attributes and how objects respond to messages,
- Define Polymorphism, Inheritance, data abstraction, encapsulation, and protocol,
- Describe objects relationships,
- Describe object persistence,

WHAT IS AN OBJECT?

- The term object was first formally utilized in the **Simula language** to **simulate some aspect of reality.**

- Attributes or properties describe object‘s state (data) and methods (properties or functions) define its behavior.

- An object is an entity.
  - It knows things (has attributes)
  - It does things (provides services or has methods
  - Examples in next Slide …….

OBJECT'S ATTRIBUTES

- Attributes represented by data type.
- They describe objects states.
- In the Car example the car's attributes are:
- color, manufacturer, cost, owner, model, etc.

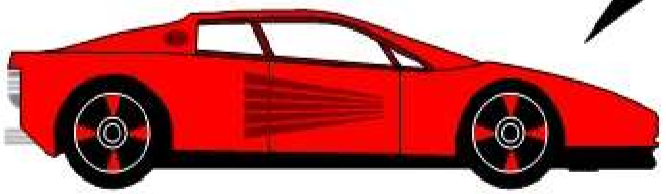
OBJECT'S METHODS

- **Methods define objects behaviour and specify the way in which an Object's data are manipulated.**
- **In the Car example the car's methods are:**
- **drive it, lock it, tow it, carry passenger in it.**

IT KNOWS THINGS (ATTRIBUTES)

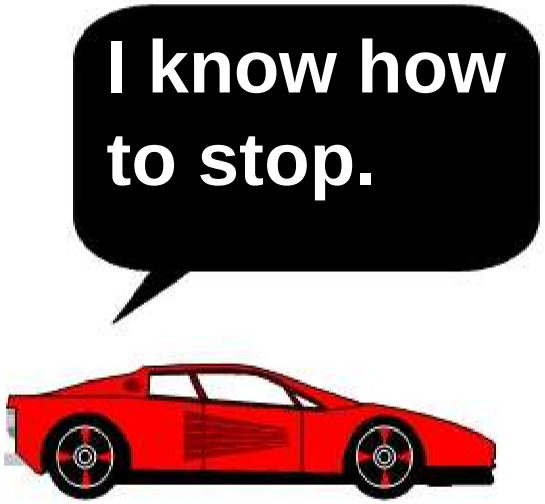
ATTRIBUTES

I am a Car.
I know my color, manufacturer, cost, owner and model.



IT DOES THINGS (METHODS)

METHODS



Object is whatever an application wants to talk about.

- For example, Parts and assemblies might be objects of bill of material application.

- Stocks and bonds might be objects of financial investment applications.

OBJECTS ARE GROUPED IN CLASSES

- The role of a class is to **define the attributes and methods** (the state and behaviour) **of its instances.**
- Used to **distinguish one type of object from the other.**
- Set of objects**, that share common methods, structure, behaviour.**
- **Single object** is simply an **instance of class.**
- The **class car**, for example, **defines the property color.** Each individual car (object) will have a **value for this property, such as "maroon," "yellow" or "white."**

Employee Class

John object **Jane object** **Mark object**