

OBJECT-ORIENTED MODELLING (OOM) - 32536

MODULE 1

**Fundamentals of Object Orientation;
Brief Introduction to the Unified
Modelling Language (UML)**
(POOA-Chapter 1; POOD-Chapter 1)

Module Outline

- Why object-orientation?
- Classes and objects
- The need for modelling
- A brief history of the Unified Modeling Language (UML)

Why Object-orientation?

- OO Software Engineering results in easier maintainability and extensibility
 - Resulting from localized and controlled changes
- OO Software Engineering results in more Opportunities for Reuse
 - Code, Design (including Patterns and Frameworks), Requirements (Use cases), Components (Link time and Run time)
- OO Software Engineering leads towards Component-based development and Service-orientation

How objects work

- Classes are design-time constructs; Objects are run-time constructs
- Objects and classes bind together in a single module all functionality and associated data
- Objects maintain their own state (values of attributes)
- System functionality is achieved at run time by objects passing messages

Shift of Mindset

- Focus on data and associated processes – think in terms of the “responsibilities” that an object might possess (responsibilities for knowing, for doing and for ensuring).

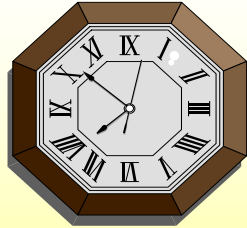
5

Fundamentals of Object Orientation

Basic Concepts

Identity, modularity, abstraction
(especially classification),
polymorphism (via inheritance)

Classes and Objects



A "CLOCK" in general
is a CLASS.
Your CLOCK and
My CLOCK are specific
Objects

What is a Class?

- A Class is a *DEFINITION*, a *TEMPLATE*, for the Objects
- Objects are *INSTANCES* of a Class
 - NOTE: A Class is *NOT* a Collection of Objects;
 - Example: A 'class' of OOA students sitting together in a lecture room are *NOT* a Class in the object-oriented sense

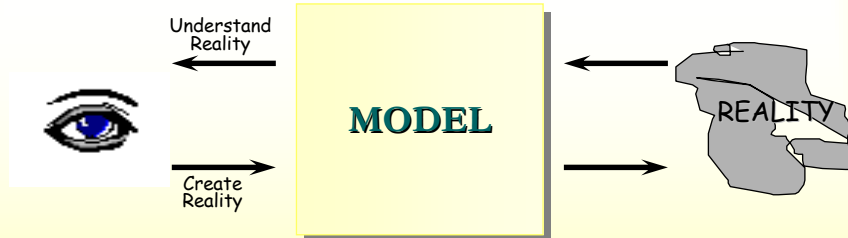
The Star of OO Fundamentals



More details in Module 2

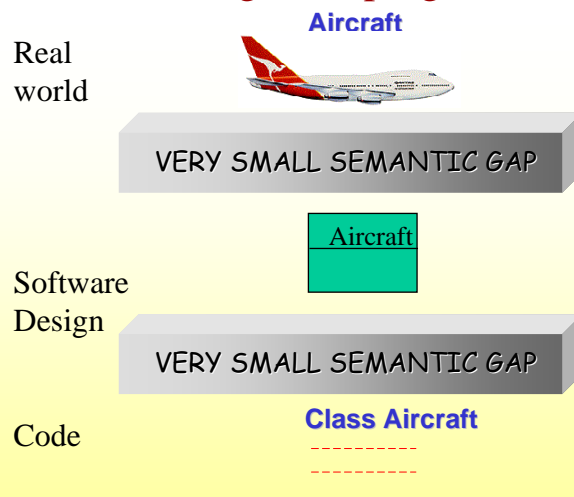
The Need for Modelling

Why model?



Understanding the existing systems, applications and processes by creating models from them, and modeling the new systems that we want to create.

OO Modelling is a bridge between the real (business) world and the software design and program.



The lifecycle is "seamless".

The Unified Modeling Language

Structure and History

13

**A modelling language consists of
a metamodel plus a notation**

A metamodel is a model of models

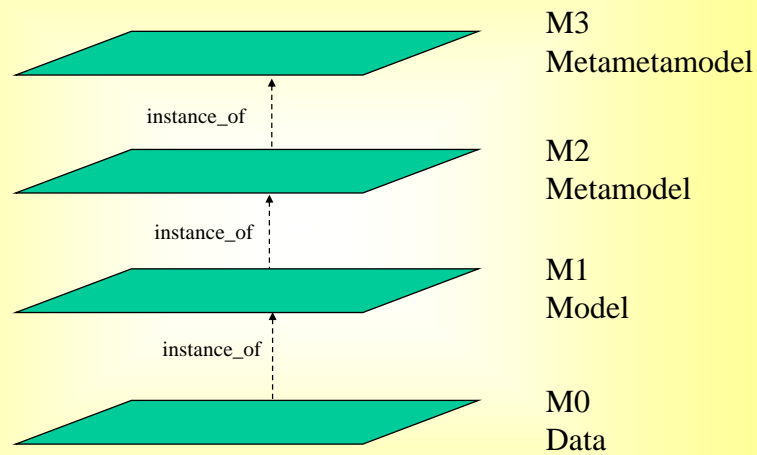
Metamodel gives rules/semantics

Notation communicates ideas

UML is one such modelling language

14

UML lies within a four layer architecture (which itself is problematical)



- UML (at level M2) is actually a metamodel which is a model of all the models that you build in designing a system. It is the set of rules you follow (and which are embodied in a CASE tool or a drawing tool like Visio).

Over the past two decades

- There have been numerous notations proposed. Some have survived, many died but have influenced current trends.
- UML was created from six submissions made in 1997 under the auspices of the Object Management Group (OMG).
- It has strong influences from the old Booch and OMT notations and some input from the OPEN Modelling Language (OML) created at UTS.

17

UML has many versions

- Version 1.1 (August 1997)
- Version 1.3 (March 2000) was stable and has many books written on it
- Version 1.4.x was stable. V1.4 in September 2001; V1.4.1 (a.k.a. 1.5) in March 2003; Version 1.4.2 (ISO/IEC 19501) in July 2004 is used primarily here.
- Version 2.0 in July 2005; 2.1.1 in February 2007; 2.1.2 in November 2007.

18

Unfortunately:

- Although an OMG standard and an ISO standard (Version 1.4.2), it has some flaws and errors in the metamodel. Watch out for these problems and develop work arounds.
- For instance, aggregation and composition are not well enough defined for everyone using UML to have the same semantics. Make sure everyone in your team is using the same semantics (especially for aggregation – white diamond notation). Also stereotypes are seriously flawed and used badly in practice.

19

Conclusions

- Modelling is needed for quality designs and for developing software
- A modelling language is needed. This consists of a notation underpinned by a metamodel
- One popular such language is UML from the OMG