Rumbaugh Methodology
Booch Methodology
Jacobson Methodology

KRISHNENDU GUHA

ASSISTANT PROFESSOR (ON CONTRACT)

NATIONAL INSTITUTE OF TECHNOLOGY (NIT), JAMSHEDPUR

EMAIL: KRISHNENDU.CA@NITJSR.AC.IN

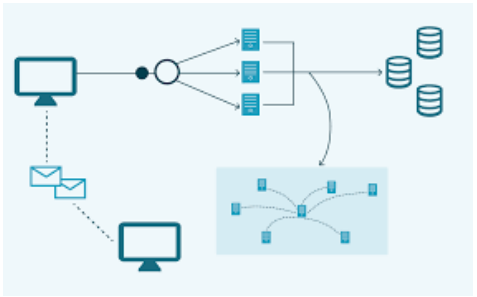
Rumbaugh Methodology

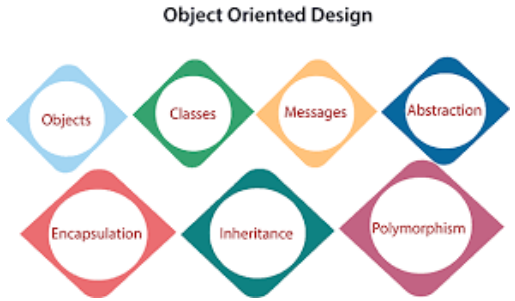
- It was developed around 1991 by Rumbaugh, Blaha, Premerlani, Eddy and Lorensen as a method to develop object-oriented systems and to support object-oriented programming
- OMT (Object Modeling Technique) describes a method for the analysis, design, and implementation of a system using an object-oriented technique.
- Class attributes, method, inheritance, and association also can be expressed easily
- Describing Object Model or the static structure of the system

James Rumbaugh

Phases of OMT

- OMT consists of four phases, which can be performed iteratively:
- Analysis -The results are objects and dynamic and functional models
- System Design - The result is a structure of the basic architecture of the system.
- Object Design - This phase produces a design document, consisting of detailed objects and dynamic and functional models Implementation-
- This activity produces reusable, extendible, and robust code



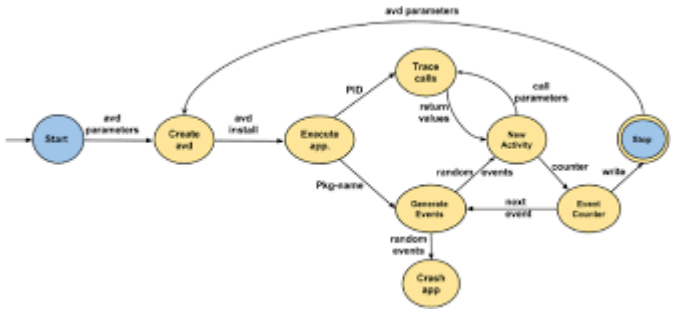


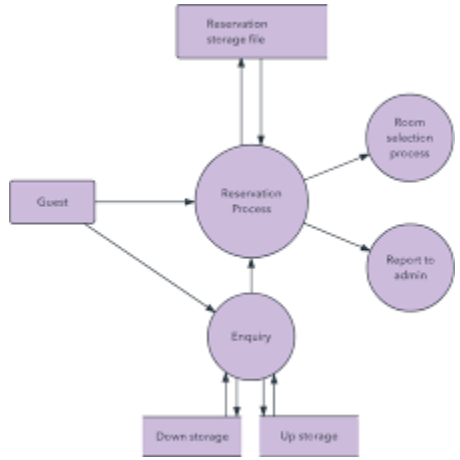
OMT Modeling

- OMT separates modeling into three different parts:
- An object model, presented by the object model and the data dictionary.
  - structure of objects in a system.
  - Identity, relationships to other objects, attributes and operations.
  - Object diagram
  - Classes interconnected by association lines
  - Classes- a set of individual objects
  - Association lines- relationship among classes (i.e., objects of one class to objects of another class)
- A dynamic model, presented by the state diagrams and event flow diagrams.
  - States, transitions, events and actions
  - OMT state transition diagram-network of states and events
- A functional model, presented by data flow and constraints

Object model	Dynamic model	Functional model
• Represents the static, structural, 'data' aspects of a system	• Represents the temporal, behavioural, 'control' aspects of a system	• Represents the transformational, 'functional' aspects of a system

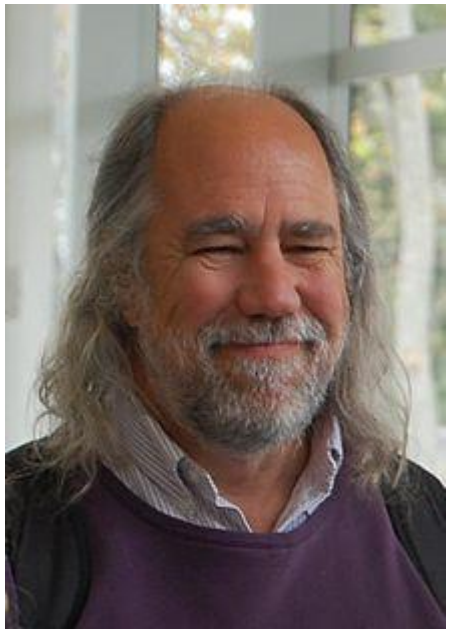
- The Rumbaugh object model is very much like an entity relationship diagram except that there are now behaviors in the diagram and class hierarchies.
- The dynamic model is a "state transition" diagram that shows how an entity changes from one state to another state.
- The functional model is the equivalent of the familiar data flow diagrams from a traditional systems analysis.





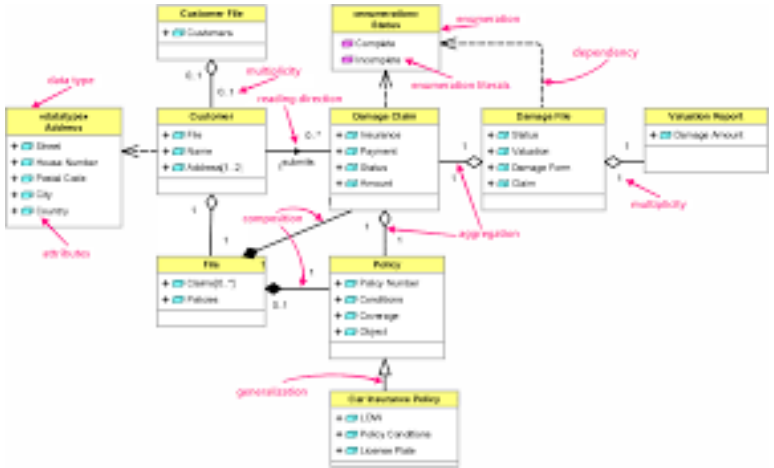
Booch method

- The **Booch method** is a method for object-oriented software development.
- It is composed of an object modeling language an iterative object-oriented development process, and a set of recommended practices
- The method was authored by Grady Booch when he was working for Rational Software (acquired by IBM), published in 1992 and revised in 1994.
- It was widely used in software engineering for object-oriented analysis and design and benefited from ample documentation and support tools.



Grady Booch

- The notation aspect of the Booch method was superseded by the Unified Modeling Language (UML), which features
- graphical elements from the Booch method along with
- elements from the object-modeling technique (OMT) and
- object-oriented software engineering (OOSE).



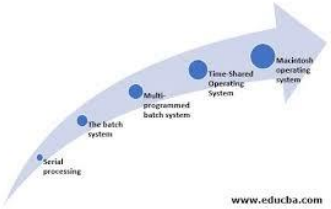
UNIFIED MODELING LANGUAGE™

Content of the method

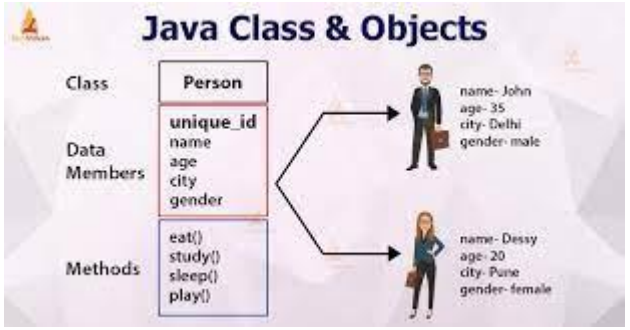
- The Booch notation is characterized by cloud shapes to represent classes and distinguishes the following diagrams:

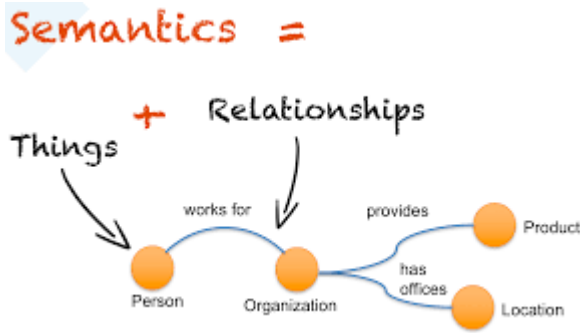
Model	Type	Diagram	UML correspondence
Logical	Static	Class diagram	Class diagram
		Object diagram	Object diagram
	Dynamic	State transition diagram	State chart diagram
		Interaction diagram	Sequence diagram
Physical	Static	Module diagram	Component diagram
		Process diagram	Deployment diagram

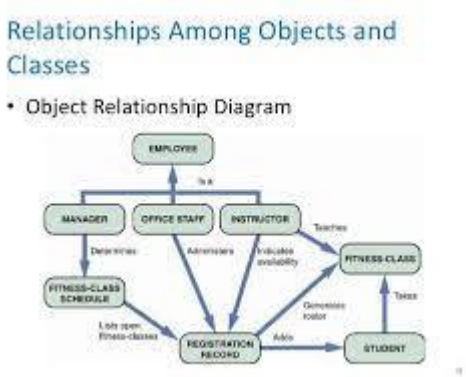
- The process is organized around a macro and a micro process
- The macro process identifies the following activities cycle:
- Conceptualization : establish core requirements
- Analysis : develop a model of the desired behavior
- Design : create an architecture
- Evolution: for the implementation
- Maintenance : for evolution after the delivery

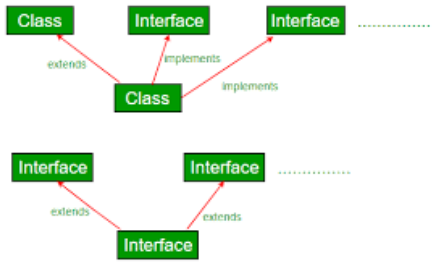


- The micro process is applied to new classes, structures or behaviors that emerge during the macro process. It is made of the following cycle:
- Identification of classes and objects
- Identification of their semantics
- Identification of their relationships
- Specification of their interfaces and implementation









Jacobson methodology

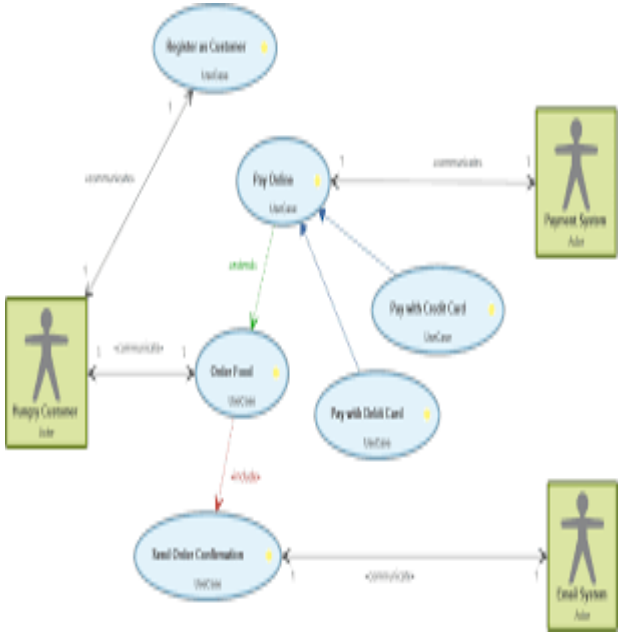
- The Jacobson et al. methodologies cover the entire life cycle and stress traceability between the different phases both forward and backward.
- (example Object – Oriented Business Engineering (OOBE), Object Oriented Software Engineering (OOSE) and Objectory)
- This traceability enables reuse of analysis and design work, possibly much bigger factors in the reduction development time then reuse of code.
- At the heart of their methodologies is the use-case concept, which evolved with Objectory (Object Factory for Software Development).

Ivar Jacobson

Use Cases

- Use cases use cases are scenarios for understanding system requirements.
- A use case is an interaction between users and the system. the use case model captures the goal of the user and the responsibility of the system to its users.
- In the requirements analysis, the use cases are described as of the following.
  - Non formal text with no clear flow of events
  - Text, easy to read but with a clear flow of events to follow
  - Formal styles using pseudo code.
- The use case description must contain
  - ***How*** and ***when*** the use case begins and ends.
  - The interaction between the use case and its actors including ***when*** the interaction occurs and ***what*** is exchanged.
  - ***How*** and ***When*** the use case will need data stored in the system or will store data in the system.
  - ***Exceptions*** to the flow of events.
  - ***How*** and ***when*** concepts of the problem domain are handled.

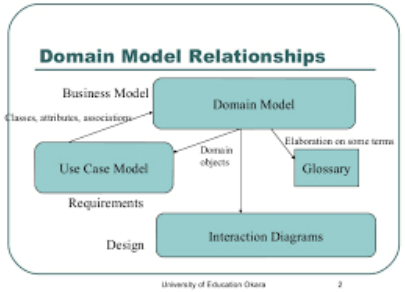
- Every single use case should describe one main flow of events.
- An exceptional or additional flow of events could be added.
- The exceptional use case extends another use case to include the additional one.
- The use-case model employs extend and uses relationships.
- The extends relationship is used when you have one use case that is like another use case but does a bit more. In essence extends the functionality of the original use case (like a subclass) .
- The user's relationship reuses common behavior in different use cases.
- Use cases would be viewed as a concrete or abstract.
- An Abstract use case is not complete and has no actors that initiated it but it is used by another use case.
- this inheritance could be used in several levels.
- abstract use cases also are the ones that have uses or extends relationship.

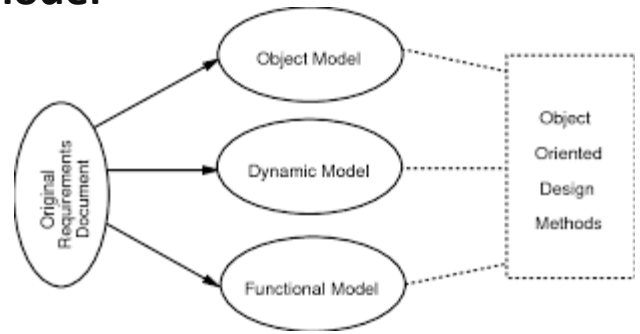


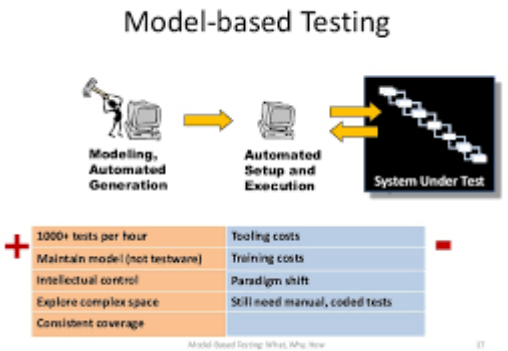
Object oriented software engineering: Objectory

- object oriented software engineering is also called ***Objectory***, is a method of object-oriented development with the specific aim to fit the development or large, real- time systems.
- The development process called use-case driven development, stresses that use cases are involved in several phases of the development including analysis, design, validation, and testing.
- The use-case scenario begins with a user of the system initiating a sequence of interrelated events.
- The system development method on OOSE, Objectory, is a disciplined process for the industrialized development of software, based on the use-case driven design that centers on understanding the ways in which the system actually is used.
- By organizing the analysis and design models around sequences of user interactions and actual usage scenarios, the method produces system that are both most usable and more robust, adapting more easy to change in usage.

- The Jacobson et al.’s Objectory has been developed and applied to numerous application areas and embodied in the CASE tools systems.
- Objectory is built around several different models:
- **use case model**
- **domain object model**
- **analysis object model**
- **Implementation model**
- **Test model**








THANKS