



Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

Module 01: Object-Oriented Analysis & Design

Challenges in Software Engineering

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Understand why Software is still *developed* and **not constructed**
- Understand why Software projects fail
- Take a glimpse of the remedial measures
- KYC: Know Your Course





Module Outline

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Is Software *Engineering*?
 - Evolution of Engineering Domains
- Why Software Projects Fail?
- Software: Development → Construction
- KYC: Know Your Course
 - Course Outline
 - Course Text Book
 - Course Examples
 - Course TAs



Software Engineering

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

Software Development



The order of the day is to refer to **Software Development** as **Software Engineering**

Is it fair?

Can we really **Construct Software** as Civil Engineers build bridges?



Engineering: Skills of Construction

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- **Civil Engineering**
 - Construction of Buildings
- **Mechanical Engineering**
 - Construction of Automobiles
- **Electrical Engineering**
 - Construction of Power Plants
- **Software Engineering**
 - *Development of Software*



Evolution of Domains

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Bridges
- Surgery
- Airplanes
- Software



Bridge Construction – Art of Connecting

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

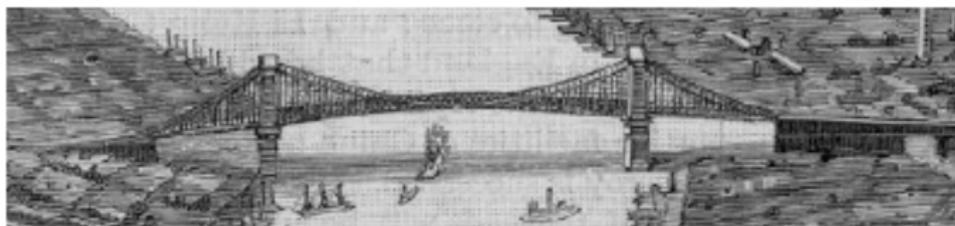
Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Early Wood, Stone – *time immemorial*
- Then Iron, Steel
- Concrete Bridges
- *Constructing a bridge* is different from *innovating a bridge* (with new material, for instance) for the first time
- *Engineers use well established metrics, standards to design bridges* - they do not innovate at this stage





Medical Surgery – Art of Curing

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

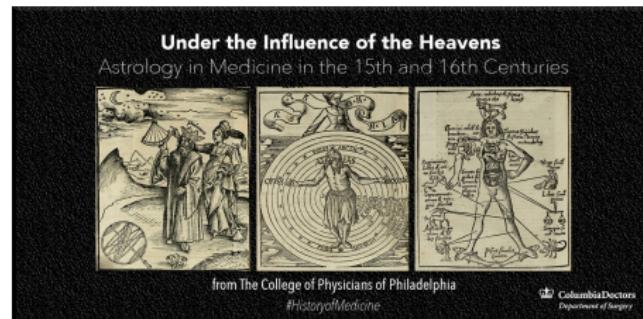
Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- *Health was thought to be restored by purging, starving, vomiting or bloodletting*
 - Surgeons and barbers specialized in this practice
 - Widely practiced in 18th and 19th century
 - Declared quackery by 1900
- Infection control
 - Survived surgery, died out of infection
 - Germ theory and sterility came only in late 1800's
 - *Strict protocols for Surgery. Rate of infection < 2.5%*





Airplanes – Art of Flying

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- 400 BC Chinese fly kite aspiring humans to fly
- For centuries, we tried to fly like birds ... disastrous
- Steam powered, hot air
- Gliders, single man
- Engine powered
- 1903 Wright brothers' first flight - 12 seconds, 120 feet, 10 feet altitude
- *Manufacture of airplanes and flying them are strictly regulated by proven practices, standards*





Software Development – Art of Problem Solving

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Vision of Mechanical Computing by Babbage (1810's)
- Relatively nascent field in comparison
 - Incompleteness Theorem (1931) by Kurt Godel
 - Turing machines (1936) by Alan Turing
- Today, machines are getting faster or more powerful
- Yet, is software delivered *successfully*? That is,
 - On time
 - Within budget
 - Feature complete
 - Working (failure free)



Success of Software Projects – Or the Lack of It

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- How successful are we in developing software?
- More than 30% of software projects fail - irrespective of development paradigm



2011 is the first year where we asked about Lean.

We only had 40 respondents for this paradigm.

Copyright 2011 Scott W. Ambler www.ambysoft.com/surveys/

Source: <http://www.ambysoft.com/surveys/success2011.html>



Success of Software Projects – Or the Lack of It

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

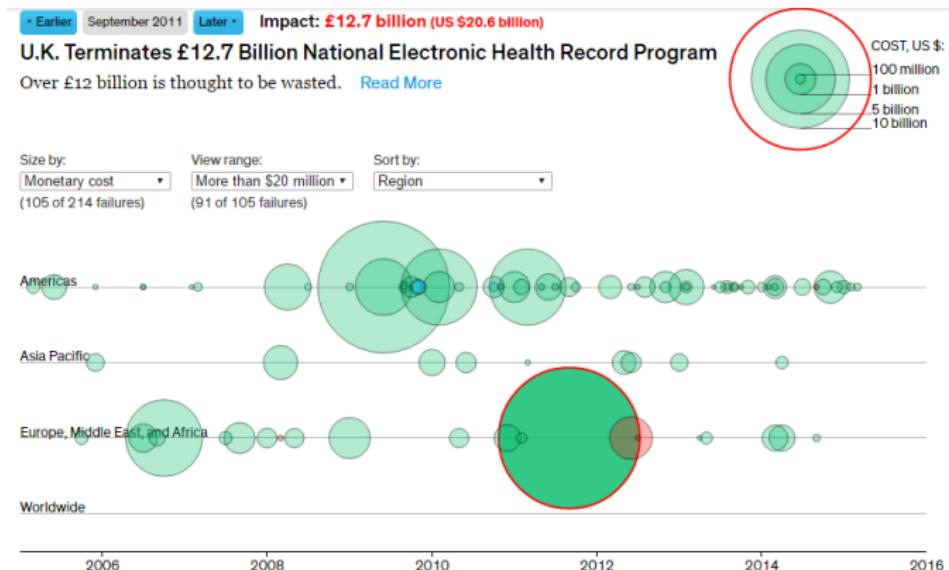
Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- A decade of failures – 2006 to 2016



Source: <http://spectrum.ieee.org/static/the-staggering-impact-of-it-systems-gone-wrong>



Why Projects Fail? – Complexity is the culprit!

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

• Complexity: Changes From Requirements

- Customers Learn from the Solution
- Business Environment and Conditions Change
- Business Processes are Re-engineered

• Complexity: Changes From Technology

- Tools/Platform Release New Versions
- Actual Tool/Platform Capabilities may Vary from Plans

• Complexity: Changes From People

- Interactions are Complex
- Individual Behavior is Unpredictable

• Complexity: ...



Software Engineering

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs
Summary

- If software engineering like manufacturing or like designing a manufacturing plant?
 - Is it like making another cell phone? Or,
 - Is it like the making of cell phones?
 - It took 37 years for commercialization
- **Manufacturing** must be *predictive*
 - Measure and control quality
 - Measure and control quantity
- **Designing a manufacturing plant** is *creative / innovative*
 - Most software development is innovative process rather than predictive manufacturing
 - Requires great deal of innovation, interaction / communication



Software Development → Software Construction

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Structured Analysis, Modeling, Design, and Implementation
- Adherence to *Good Practices*

Object-Oriented Analysis & Design is a precursor to both



Object-Oriented Analysis and Design: Course Outline

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- **Software Complexity:** Understanding the challenges OOAD can address
- **Object Model:** Defining the primitives of the OO paradigm
- **Classes and Objects:** Bringing in the broader perspectives
- **Classes and Objects:** Identification approaches using OOAD
- **Unified Modeling Language:** Understanding UML Diagrams (3 weeks)
- **OOAD Case Studies:** Applying OOAD in different contexts



Text Book

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

Object-Oriented Analysis and Design – With Applications
by *Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. Houston*, 3rd Ed., 2007



Example Systems for OOAD

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- Running Example in Lectures
 - **Leave Management System (LMS)**
 - Examples from Booch's OOAD Book
- Running Example in Assignments
 - **(Students') Assignment Management System (AMS)**
 - Examples from Booch's OOAD Book
- Examples for Complete Workout
 - **(Indian) Postal Management System (PMS):**
 - **(Newspaper) Story Management System (SMS)**
 - **(Rental) Car Management Systems (CMS)**
 - Examples from Booch's OOAD Book



Instructor and TAs

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module Summary

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

- *Software Engineering* is closer to *Development* than *Construction*
- Software Projects fails for Complexity, Change
- **OOAD** can take software from *Development* to *Construction*



Instructor and TAs

Module 01

Partha Pratim
Das

Objectives &
Outline

Is Software
Engineering?

Evolution of
Domains

Why Software
Project Fail?

Software:
Development
→
Construction

KYC
Outline
Text
Examples
TAs

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity
Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

Module 02: Object-Oriented Analysis & Design

Complexity of Software

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Understand why Software is Complex
- Study the elements of Complexity



Module Outline

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Defining Software Complexity
- Four Elements of Complexity
 - The Complexity of the Problem Domain
 - The Difficulty of Managing the Development Process
 - The Flexibility Possible through Software
 - The Problems of Characterizing the Behavior of Discrete Systems



Module 02: Lecture 02

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Defining Software Complexity
- Four Elements of Complexity
 - The Complexity of the Problem Domain
 - The Difficulty of Managing the Development Process



What do we refer to as *Software*?

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

- Personal or Limited-use Software
- Industrial-Strength Software



Personal or Limited-use Software

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

- Limited set of behaviors
- Not very complex
- Specified, constructed, maintained, and used by the same person or a small group
 - Amateur programmer, or
 - Professional developer working in isolation
- Tend to have short life span
- Can be thrown away and replaced with entirely new software rather than attempt to reuse them, repair them, or extend their functionality
- Generally more tedious than difficult to develop
- Techniques to develop them is of little interest here



Industrial-Strength Software

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Exhibit a very rich set of behaviors – *reactive systems that drive or are driven by events in the physical world*
- Works with scarce resources – *time, space, power, ...*
- Maintains the integrity of millions of records while allowing concurrent updates and queries – *airline booking*
- Commands and controls of real-world entities – *routing of air or railway traffic*
- Tend to have a long life span
- Depended by many users over time on proper functioning
- Usually based on frameworks that simplify the creation of domain-specific applications

Complexity of Industrial-Strength Software systems exceeds the human intellectual capacity



Software is Inherently Complex

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

Inherent complexity derives from four elements:

- The Complexity of the Problem Domain
- The Difficulty of Managing the Development Process
- The Flexibility Possible through Software
- The Problems of Characterizing the Behavior of Discrete Systems



The Complexity of the Problem Domain

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- **Domains are difficult to understand.** For example:
 - Electronic system of a multi-engine aircraft
 - Merchant Shipping
 - Online Trading and Reconciliation
- **Functional Requirements** are
 - *Complex to master*
 - Often are competing, even contradictory
- **Non-Functional Requirements** (*usability, performance, cost, survivability, and reliability*) are
 - *Often Implicit*
 - Difficult to justify in budget



The Complexity of the Problem Domain

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain

Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- **Communication Gap** between *Users* and *Developers*
 - Users *cannot express*; developers *cannot understand*
 - *Lack of expertise* across domains
 - *Different perspectives* on the nature of the problem leading to *different assumptions* regarding the nature of the solution
 - *Few instruments* to precisely capture requirements – large texts with some drawings is all we have
 - Leads to *External Complexity*
- **Changing / Evolving Requirements** during Development
 - Early products lead to *better understanding of needs* by the users
 - Developers get *enlightened through the process of development*



The Complexity of the Problem Domain

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

● Large Capital Investments

- Make *scrapping of projects unfeasible* with changing requirements
- Lead to *inordinate percentage of software preservation*

Maintenance	<i>Correct errors</i>
Evolution	<i>Respond to changing requirements</i>
Preservation	<i>Use extraordinary means to keep an ancient and decaying piece of software in operation</i>



The Difficulty of Managing the Development Process

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Create **Illusion of Simplicity** to hide external complexity.
Hence code bases get large
- Code bases are **large in size** (LOC: 10^5 to 10^6 & more)
 - In spite of the *use of smart languages*
 - In spite of the *reuse of designs and codes*
 - Goes well beyond the comprehension of a single (or small group of) individual/s – even with meaningful decomposition. Hence, we need teams
- Challenges associated with **team development**
 - Large code bases mean more teams with *more developers in more geographies*
 - More developers means *more complex communication* and hence *more difficult coordination*
 - Key management challenge is to maintain a *unity and integrity of design*



Module 02: End of Lecture 02

- Defining Software Complexity
- Four Elements of Complexity
 - The Complexity of the Problem Domain
 - The Difficulty of Managing the Development Process

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary



Module 02: Lecture 03

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Four Elements of Complexity
 - The Flexibility Possible through Software
 - The Problems of Characterizing the Behavior of Discrete Systems



The Flexibility Possible through Software

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

Civil Construction

- Sources its material (timber, steel, cement, glass, etc.) from other vendors
- Has **uniform building codes** and standards for the quality of raw materials
- Rarely would a builder think about **adding a new sub-basement** to an existing 100-story building

Software Development

- Often **builds the components** within team – because components are also software and can be built
- **Few standards** exist for reusable components
- Amazingly, users of software systems rarely think twice about **asking for equivalent changes**



The Problems of Characterizing the Behavior of Discrete Systems

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process

Flexibility of
Software

Behavior of
Discrete Systems

Summary

- Software is built and executed on digital computers – Hence, they are **Discrete Systems**
- A large application would have
 - hundreds or even thousands of variables
 - more than one (often several) thread of control
- A state is:
 - Entire collection of variables
 - The current values of variables
 - The current address of each process
 - The calling stack of each process

A Software has finite number of states. Yet, many of these the states are often intractable and influenced by external factors



Module Summary

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

- Understood the difference between Limited-use and Industry-strength software
- Studied the four elements of Software Complexity



Instructor and TAs

Module 02

Partha Pratim
Das

Objectives &
Outline

Defining
Software
Complexity

Elements of
Complexity

Problem Domain
Development
Process
Flexibility of
Software
Behavior of
Discrete Systems

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

Module 03: Object-Oriented Analysis & Design

Structure and Attributes of a Complex System

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

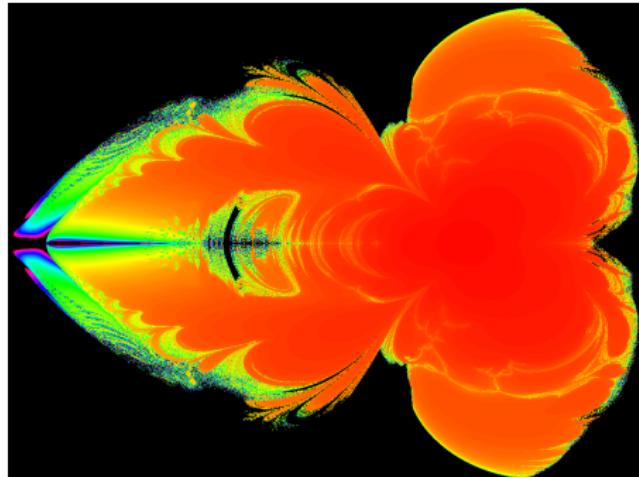
Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Understand the Structure of Complex Systems with examples of Man-made, Natural, and Social Systems
- Understand the Attributes of Complex Systems





Module Outline

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns

Stable Forms

Summary

- Structure of Complex Systems – Examples

- Personal Computer
- Plants
- Education System in India

- Attributes of a Complex System

- Hierarchic Structure
- Relative Primitives
- Separation of Concerns
- Common Patterns
- Stable Intermediate Forms



Module 03: Lecture 04

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer

Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure

Relative
Primitives

Separation of
Concerns

Common
Patterns

Stable Forms

Summary

- Structure of Complex Systems – Examples
 - Personal Computer
 - Plants
 - Education System in India
- Attributes of a Complex System
 - Hierarchic Structure



The Structure of a Personal Computer

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Central Processing Unit (CPU) – *Executes programs*
 - Arithmetic & Logic Unit (ALU)
 - Registers
 - NAND gate
 - * CMOS Gate
 - * Interconnect
 - Inverter
 - Control Logic
 - Primary Memory
 - Bus – peripheral devices
- Hard disk drive – *Persistent storage for data*
- Monitor – *Outputs data*
- Keyboard – *Inputs data*
- Secondary storage device (DVD drive / USB) –
Removable storage for data



Building Blocks

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Personal Computer
 - Gates – NAND, Inverter, etc.
 - Interconnections



Complex Systems are Hierarchic

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a

Complex
System

Hierarchic
Structure

Relative
Primitives

Separation of
Concerns

Common
Patterns

Stable Forms

Summary

- Each Level of Hierarchy represents a Layer of Abstraction
- Each Layer
 - Is built on top of other layers: CMOS Gates → NAND Gates→ Registers
 - (in turn) Supports other layers: CMOS Gates → NAND Gates→ Registers
 - Is independently understandable: CPU
 - Works independently with clear separation of concerns: ALU, Memory
- Common services / properties are shared across Layers: Same power-tree feeds the components of CPU
- Layers together show **Emergent Behavior**
Behavior of the whole is greater than the sum of its parts
- Systems demonstrate cross-domain commonality: Cars have processors, memory, display



The Structure of Plants

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Roots – *Absorb water and minerals from the soil*
 - Branch Roots
 - Collection of Cells
 - Chloroplasts
 - Nucleus
 - Root Hairs
 - Collection of Cells ...
 - Root Apex
 - Root Cap
 - Stems – *Transport raw materials from roots up to leaves*
 - Leaves – *Use the water and minerals for photosynthesis*
 - Epidermis
 - Mesophyll
 - Vascular Tissue



Building Blocks

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Plants

- Cells – different types
- Vessels



Complex Systems are Hierarchic

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Each Level of Hierarchy represents a Layer of Abstraction
- Each Layer
 - Is built on top of other layers: Cells → Branch Roots → Roots
 - (in turn) Supports other layers: Cells → Branch Roots → Roots
 - Is independently understandable: Leaf
 - Works independently with clear separation of concerns: Roots, Leaves
- Common services / properties are shared across Layers: Oxygen supply to Roots, Stems, & Leaves
- Layers together show **Emergent Behavior**
Behavior of the whole is greater than the sum of its parts
- Systems demonstrate cross-domain commonality: Cells are constituents of plants & animals



The Structure of Education System in India

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- CFTIs – IIT, IIM, NIT, IISER, etc.
 - Departments
 - Students
 - Teachers
 - Staffs
 - Library
- UGC / AICTE – *Regulates higher education*
 - Universities
 - Colleges
 - Departments
 - * Students ...
- CBSE / ICSE – *Regulates school education*
 - Schools
 - Students ...
 - Play Ground



Building Blocks

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

● Education System

- Knowledge Delivery – Contact, Remote, MOOCs
- Examination – Written, Oral, Practical
- Admission
- Graduation



Complex Systems are Hierarchic

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Each Level of Hierarchy represents a Layer of Abstraction
- Each Layer
 - Is built on top of other layers: Departments → Colleges → Universities
 - (in turn) Supports other layers: Departments → Colleges → Universities
 - Is independently understandable: College
 - Works independently with clear separation of concerns: Departments, Library
- Common services / properties are shared across Layers: NKN is shared between institutes; Departments use the same telephone system
- Layers together show **Emergent Behavior**
Behavior of the whole is greater than the sum of its parts
- Systems demonstrate cross-domain commonality: Similar Leave policy for Teachers & Staff of PSUs



Building Blocks – RECAP

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Personal Computer
 - Gates – NAND, Inverter, etc.
 - Interconnections
- Plants
 - Cells – different types
 - Vessels
- Education System
 - Knowledge Delivery – Contact, Remote, MOOCs
 - Examination – Written, Oral, Practical
 - Admission
 - Graduation



Complex Systems are Hierarchic – RECAP

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Each Level of Hierarchy represents a Layer of Abstraction
- Each Layer
 - Is built on top of other layers: CMOS Gates → NAND Gates→ Registers; Cells → Branch Roots → Roots; Departments → Colleges → Universities
 - (in turn) Supports other layers: CMOS Gates → NAND Gates→ Registers; Cells → Branch Roots → Roots; Departments → Colleges → Universities
 - Is independently understandable: CPU, Leaf, College
 - Works independently with clear separation of concerns: ALU, Memory; Roots, Leaves; Departments, Library
- Common services / properties are shared across Layers: Same power-tree feeds the components of CPU; Oxygen supply to Roots, Stems, & Leaves; NKN is shared between institutes; Departments use the same telephone system
- Layers together show **Emergent Behavior**
Behavior of the whole is greater than the sum of its parts
- Systems demonstrate cross-domain commonality: Cars have processors, memory, display; Cells are constituents of plants & animals; Similar Leave policy for Teachers & Staff of PSUs; Laws of conservation of momentum apply equally in astronomy (macro) and nuclear physics (micro)



Complex Systems are Hierarchic – RECAP

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants

Education
System in India

Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Each Level of Hierarchy represents a Layer of Abstraction
- Each Layer
 - Is built on top of other layers
 - (in turn) Supports other layers
 - Is independently understandable
 - Works independently with clear separation of concerns
- Common services / properties are shared across Layers
- Layers together show **Emergent Behavior**
Behavior of the whole is greater than the sum of its parts
- Systems demonstrate cross-domain commonality



The Five Attributes of a Complex System

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Hierarchic Structure
- Relative Primitives
- Separation of Concerns
- Common Patterns
- Stable Intermediate Forms



Hierarchic Structure

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure

Relative
Primitives

Separation of
Concerns

Common
Patterns

Stable Forms

Summary

- All **systems** are composed of **interrelated sub-systems**
- Sub-systems are composed of sub-sub-systems, and so on
- Lowest level sub-systems are composed of **elementary components**
- All systems are parts of larger systems
- The value added by a system must come from the relationships between the parts, not from the parts per se

We can understand only those systems that have a hierarchic structure



Module 03: End of Lecture 04

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer

Plants

Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure

Relative
Primitives

Separation of
Concerns

Common
Patterns

Stable Forms

Summary

- Structure of Complex Systems – Examples
 - Personal Computer
 - Plants
 - Education System in India
- Attributes of a Complex System
 - Hierarchic Structure



Module 03: Lecture 05

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Attributes of a Complex System

- Hierarchic Structure
- Relative Primitives
- Separation of Concerns
- Common Patterns
- Stable Intermediate Forms



Relative Primitives

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- **Subjective Choice** – strongly dependent on the experience and expertise of the designer
- What is primitive for one observer may be at a much higher level of abstraction for another.

The choice of what components in a system are primitive is relatively arbitrary and is largely up to the discretion of the observer of the system



Separation of Concerns

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Hierarchic systems are:
 - **decomposable** – *can be divided into identifiable parts*
 - **nearly decomposable** – *the parts are not completely independent*

Intracomponent linkages

- Involves the *internal structure* of the components
- *Stronger*
- *High-frequency dynamics*

Intercomponent linkages

- Involves *interaction among components*
 - *Weaker*
 - *Low frequency dynamics*

Difference between intra- and intercomponent interactions provides a clear Separation of Concerns among the various parts of a system – helps the analysis and design in isolation



Common Patterns

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives

Separation of
Concerns

Common
Patterns

Stable Forms

Summary

- Complex systems have **Common Patterns**
- Complex systems are composed of only a few different kinds of subsystems in various combinations and arrangements (cells found in both plants and animals etc.)

Common Patterns are a major source of reuse in OOAD.

Examples include Design Patterns, STL in C++, Data Structures in Python etc.



Stable Intermediate Forms

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- It is extremely difficult to design a complex system correctly in one go
- Start with a simple system and then refine (*Iterative Refinement*)
- Objects, once considered complex, become the primitive objects on which more complex systems are built
- The system matures from one intermediate form to the next

A complex system that works is invariably found to have evolved from a simple system that worked

A complex system designed from scratch never works and cannot be patched up to make it work



Module Summary

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

- Analyzed the structure of man-made, natural and social complex systems to understand their generic principles by elucidation
- Summarized the attributes of a complex system:
 - Hierarchic Structure
 - Relative Primitives
 - Separation of Concerns
 - Common Patterns
 - Stable Intermediate Forms



Instructor and TAs

Module 03

Partha Pratim
Das

Objectives &
Outline

Structure of
Complex
Systems

Personal
Computer
Plants
Education
System in India
Recap

Attributes of a
Complex
System

Hierarchic
Structure
Relative
Primitives
Separation of
Concerns
Common
Patterns
Stable Forms

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

Module 04: Object-Oriented Analysis & Design

Handling of Complexity

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- Understand how to handle complexity



Module Outline

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- **Organized Complexity:** Canonical Form
 - Decomposition Hierarchy
 - Abstraction Hierarchy
 - Class and Object Structure
 - Canonical Form
- **Disorganized Complexity:** Limited Human Capacity



Handling Complexity

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- **Organized Complexity:** The Canonical Form of a Complex System
- **Disorganized Complexity:** The Limitations of the Human Capacity for Dealing with Complexity



Hierarchies of a Complex System

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- Hierarchies are of multiple types
 - Decomposition – **part-of** or **HAS-A**
 - Abstraction – **IS-A**



Decomposition (**HAS-A**) Hierarchy

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form
Decomposition
Hierarchy
Abstraction
Hierarchy
Class and Object
Structures
Canonical Form

Human
Capacity

Summary

- Personal Computer is composed of
 - CPU
 - Memory
 - Keyboard
 - HDD
 - ...
- This is **Decomposition Hierarchy**



Abstraction (IS-A) Hierarchy

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- A CPU can be one of
 - Pentium
 - Pentium + MMX
 - Pentium II
 - Celeron
 - Pentium III
 - ...
- All processors are different in details, but share the same functionality of a CPU. We say:
 - Pentium IS-A CPU
 - Pentium + MMX IS-A Pentium
 - Celeron IS-A Pentium II
 - ...
- This is **Abstraction Hierarchy**



Class and Object Structures

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form
Decomposition
Hierarchy
Abstraction
Hierarchy
Class and Object
Structures
Canonical Form

Human
Capacity

Summary

- Hierarchies are referred to as:
 - **Class Structure:** Abstraction (IS-A)
 - **Object Structure** Decomposition (HAS-A)

Note:

- *These are class structure and object structure at a higher level of abstraction*
- *These make up complex systems, for example, a jet engine, an airframe, the various types of seats, an autopilot subsystem, and so forth*
- *These are not classes and objects that we create when coding in software*



Key Hierarchies of Complex Systems

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

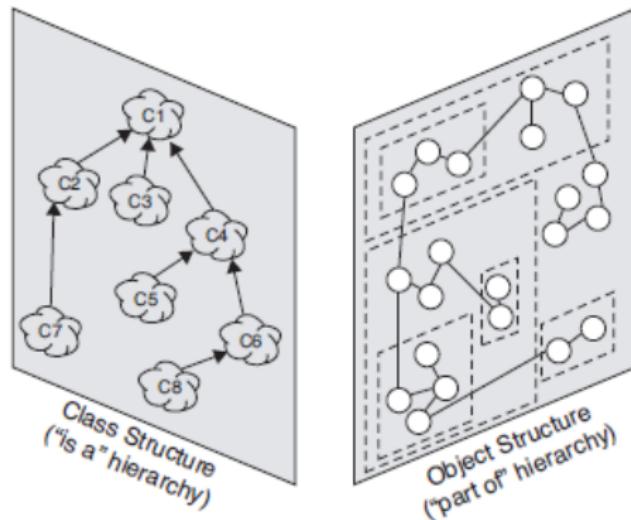
Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary



The Key Hierarchies of Complex Systems

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



The Canonical Form of a Complex System

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form
Decomposition
Hierarchy
Abstraction
Hierarchy
Class and Object
Structures
Canonical Form

Human
Capacity

Summary

- System Architecture
 - **Class Structure:** Abstraction (IS-A)
 - **Object Structure** Decomposition (HAS-A)
- Five Attributes of a Complex System
 - Hierarchic Structure
 - Relative Primitives (multiple levels of abstraction)
 - Separation of Concerns
 - Common Patterns
 - Stable Intermediate Forms
- This is **Organized Complexity**

Virtually, all complex systems take on the above canonical form



The Canonical Form of a Complex System

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

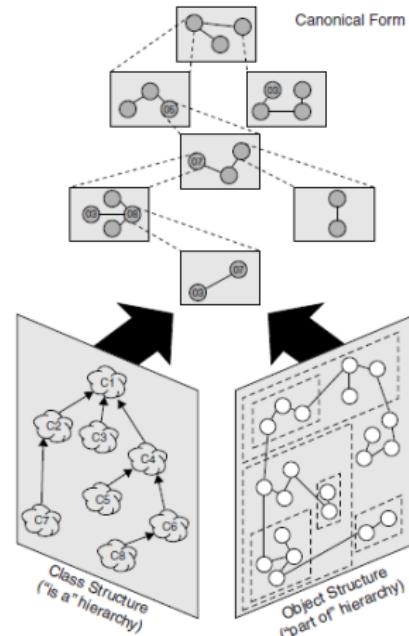
Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary



The Canonical Form of a Complex System

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



The Limitations of the Human Capacity for Dealing with Complexity

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- Maximum number of chunks of information that an individual can simultaneously comprehend is on the order of seven, plus or minus two
- Human channel capacity is related to the capacity of short-term memory
- Processing speed is a limiting factor – it takes the mind about five seconds to accept a new chunk of information
- This leads to **Disorganized Complexity**



Module Summary

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

- Outlined approaches to handling Complexity
 - **Organized Complexity:** Canonical Form
 - **Disorganized Complexity:** Limited Human Capacity

Fundamental dilemma:

The complexity of the software systems we are asked to develop is increasing, yet there are basic limits on our ability to cope with this complexity

How then do we resolve this predicament?



Instructor and TAs

Module 04

Partha Pratim
Das

Objectives &
Outline

Handling
Complexity

Canonical
Form

Decomposition
Hierarchy

Abstraction
Hierarchy

Class and Object
Structures

Canonical Form

Human
Capacity

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

Module 05: Object-Oriented Analysis & Design

Bringing Order to Chaos

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Recap

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

- **Module 01:** Why Software Projects Fail?
- **Module 02:** Four Elements of Complexity
 - The Complexity of the Problem Domain
 - The Difficulty of Managing the Development Process
 - The Flexibility Possible through Software
 - The Problems of Characterizing the Behavior of Discrete Systems
- **Module 03:** Structure and Attributes of a Complex System
 - Hierarchic Structure
 - Relative Primitives
 - Separation of Concerns
 - Common Patterns
 - Stable Intermediate Forms
- **Module 04:** Handling of Complexity
 - Decomposition Hierarchy
 - Abstraction Hierarchy
 - Class and Object Structure
 - Organized Complexity: Canonical Form
 - Disorganized Complexity: Limited Human Capacity



Module Objectives

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Build strategies for design of complex systems
- Understand design and model



Module Outline

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Bringing Order to Chaos
 - The Role of Decomposition
 - The Role of Abstraction
 - The Role of Hierarchy
- Designs and Models
 - What is Design?
 - Model Building



Module 05: Lecture 07

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Bringing Order to Chaos
 - The Role of Decomposition



The Role of Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Algorithmic Decomposition
- Object-Oriented Decomposition
- Algorithmic versus Object-Oriented Decomposition



Algorithmic Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Top-down structured design
- Divide-and-Conquer
- Decompose the problem into key processing steps



Algorithmic Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

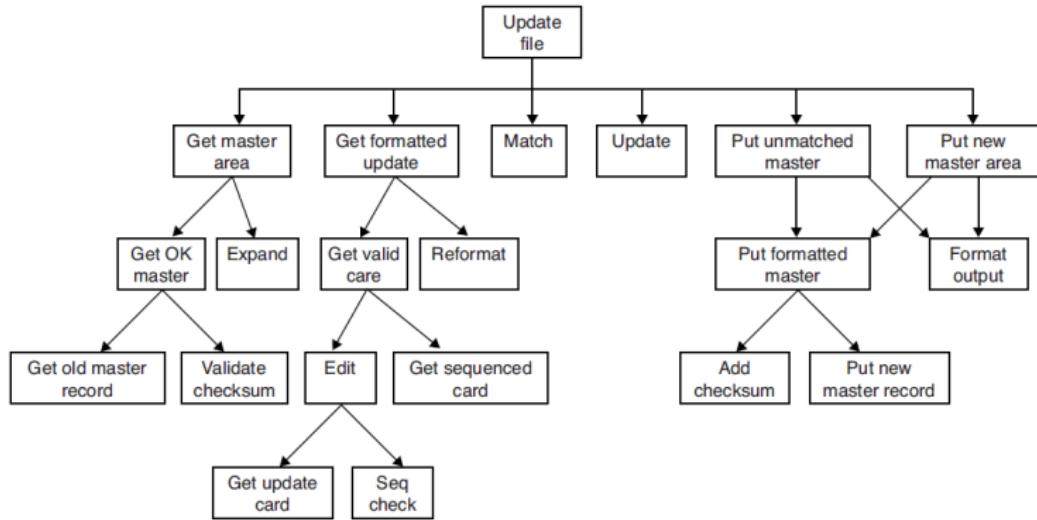
Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary



Algorithmic Decomposition: Structure Chart

- Design of a program that updates the content of a master file
- Automatically generated from a data flow diagram by an expert system tool
- Decompose the problem into steps like *Get formatted update* & *Add checksum*

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Object-Oriented Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Decompose the system according to the key abstractions in the problem domain
- Objects identified directly from the vocabulary of the problem domain



Object-Oriented Decomposition

Module 05

Partha Pratim
Das

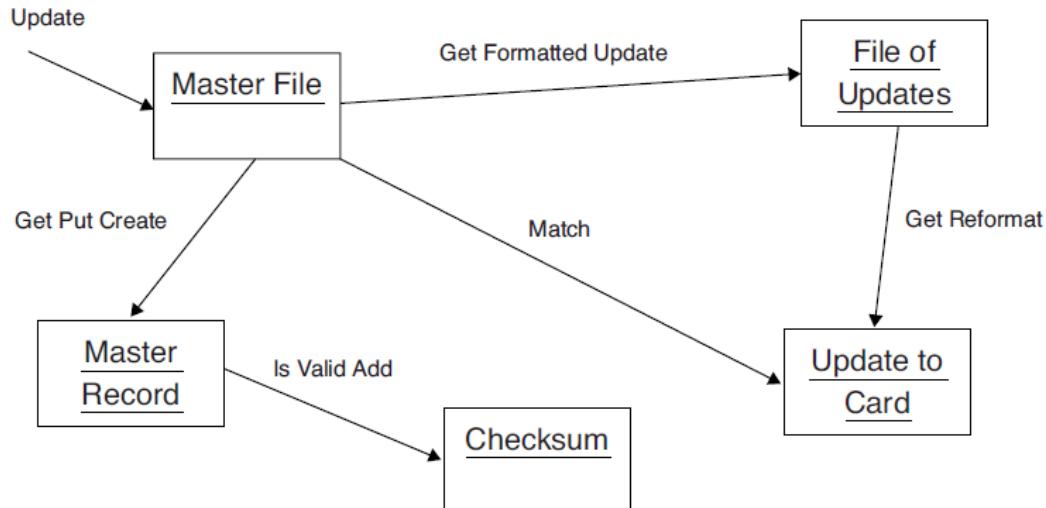
Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary



Object-Oriented Decomposition

- Decompose according to the key abstractions
- Identify objects like *Master File* and *Checksum* directly from the problem

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Object-Oriented Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Distinct approach compared to Algorithmic Decomposition
- We view the world as a set of autonomous agents that collaborate to perform some higher-level behavior
- Get Formatted Update
 - Removed as an independent algorithm
 - Becomes an operation associated with the object **File of Updates**
 - Calling Get Formatted Update on File of Updates the operation creates another object, **Update to Card**
- Each object embodies its own unique behavior, and each one models some object in the real world



Object-Oriented Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- An object is simply a tangible entity that exhibits some well-defined behavior
- Objects do things, and we ask them to perform what they do by sending them messages
- Because our decomposition is based on objects and not algorithms, we call this an **object-oriented decomposition**



Client-Server Computing Model

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

- Behavior is
 - Services provided by an object
 - Services are invoked by
 - Sending Messages
- In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



Module 05: End of Lecture 07

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Bringing Order to Chaos
 - The Role of Decomposition



Module 05: Lecture 08

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Bringing Order to Chaos
 - The Role of Decomposition – *continued*
 - The Role of Abstraction
 - The Role of Hierarchy
- Designs and Models
 - What is Design?
 - Model Building



Algorithmic versus Object-Oriented Decomposition: Evolution of Design Methods

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

- **Top-down structured design**

- Oldest and most traditional
- Does not address the issues of data abstraction and information hiding
- Does not provide an adequate means of dealing with concurrency
- Does not scale up well for extremely complex systems
- Is largely inappropriate for use with object-based and object-oriented programming languages.

- **Data-driven design**

- Mapping system inputs to outputs derives the structure of a software system
- Has been successfully applied to a number of complex domains (*information management systems*)
- Less effective for time-critical events

- **Object-oriented design**

- Models software systems as collections of cooperating objects
- Treats individual objects as instances of a class within a hierarchy of classes
- OOAD directly reflects the topology of high-order programming languages (C++ / Java)



Algorithmic versus Object-Oriented Decomposition

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

Algorithmic

- Highlights the **ordering of events**
- Typically **larger systems**
- Exposes **lower reuse**
- **Less resilient** and **more risky** to build complex systems with
- Typically flat and unable to reduce complexity
- Computing Model is typically **von Neumann**
- Low **concurrency**

Object-Oriented

- Emphasizes the **agents** that either (1) Cause action (**Clients**) or (2) Are the subjects on which these operations act (**Servers**)
- Yields **smaller systems** through the reuse of common mechanisms
- Leverages **high reuse**
- **More resilient** to change due to underlying **stable intermediate forms**
- **Reduces the risk** of building complex software systems because they evolve incrementally (**iterative refinement**)
- Directly addresses the inherent complexity of software by using the **separation of concerns** in a large state space
- Computing Model is **Client-Server**
- Inherently **distributed**; High **concurrency**



The Role of Abstraction

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- *Disorganized Complexity* results from
 - *Storage (STM) limitations* of human brain – an individual can simultaneously comprehend of the order of seven, plus or minus two chunks of information
 - *Speed limitations* of human brain – it takes the mind about five seconds to accept a new chunk of information
- **Abstraction** provides the major tool to handle Disorganized Complexity by *chunking information*
- Ignore its inessential details, dealing only with the generalized, idealized model of the object

Consider: A binary number **110010101001**

Hard to remember. Right?

Try the octal form: **(110)(010)(101)(001)** \Rightarrow **6251**

Or the hex form: **(1100)(1010)(1001)** \Rightarrow **CA9**



The Role of Hierarchy

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

- Increase the semantic content of individual chunks of information by explicitly recognizing the class and object hierarchies
- *Object structure* ⇒ Illustrates how different objects collaborate with one another through patterns of interaction that we call mechanisms
- *Class structure* ⇒ Highlights common structure and behavior within a system



What is Design?

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

Before forging ahead and plunging into Object-Oriented Design Paradigm, let us provide a crisp view of what we consider a *Design*. A Design:

- Satisfies a given (perhaps informal) functional specification
- Conforms to limitations of the target medium
- Meets implicit or explicit requirements on performance and resource usage
- Satisfies implicit or explicit design criteria on the form of the artifact
- Satisfies restrictions on the design process itself, such as its length or cost, or the tools available for doing the design



Model Building

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

- Physics
 - Time-Distance Equation
- Chemistry
 - Valency-Bond Structures
- Geography
 - Maps
 - Projections

- Electrical Circuits
 - Kirchoff's Loop Equations
 - Time Series Signals and FFT
 - Transistor Models
 - Schematic Diagram
 - Interconnect Routing
- Building & Bridges
 - Drawings – Plan, Elevation, and Side view
 - Finite Element Models

- Models are common in all engineering disciplines
- Model building follows principles of decomposition, abstraction, and hierarchy
- Each model describes a specific aspect of the system
- Build new models upon old proven models



Module Summary

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models

Design
Models

Summary

- Decomposition, Abstraction and Hierarchy can help in containing the complexity of a complex software system
- The products of design are models that enable us to reason about our structures, make trade-offs when requirements conflict, and in general, provide a blueprint for implementation
- Models help us immensely to reason with, design, build and debug systems
- We next look into **Object Models**



Instructor and TAs

Module 05

Partha Pratim
Das

Objectives &
Outline

Order to
Chaos

Decomposition
Abstraction
Hierarchy

Designs &
Models
Design
Models

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

Module 06: Object-Oriented Analysis & Design

Evolution of Object Models - Programming Languages & Paradigms

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Recap

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- **Module 01: Why Software Projects Fail?**
- **Module 02: Four Elements of Complexity**
 - The Complexity of the Problem Domain
 - The Difficulty of Managing the Development Process
 - The Flexibility Possible through Software
 - The Problems of Characterizing the Behavior of Discrete Systems
- **Module 03: Structure and Attributes of a Complex System**
 - Hierarchic Structure
 - Relative Primitives
 - Separation of Concerns
 - Common Patterns
 - Stable Intermediate Forms
- **Module 04: Handling of Complexity**
 - Decomposition Hierarchy
 - Abstraction Hierarchy
 - Class and Object Structure
 - Organized Complexity: Canonical Form
 - Disorganized Complexity: Limited Human Capacity
- **Module 05: Bringing Order to Chaos**
 - The Role of Decomposition
 - The Role of Abstraction
 - The Role of Hierarchy
 - Notion of Design and Model Building



Module Objectives

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Understand the Evolution of High-level Programming Languages through a study of Generations of Programming Languages
- Identify the role of OOP in generations



Module Outline

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Computation
 - Programming
 - Hardware Architecture
- Generations of Programming Languages with typical Topology
 - First
 - Second
 - Third
 - Fourth
 - OOP
 - Frameworks



Programming

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Programming is the process of taking an **algorithm** and encoding it into a notation
- Algorithms describe the solution to a problem in terms of the **data** needed to represent the problem
- Programs comprise of:
 - Data
 - Algorithm / Computation



Hardware Architecture

Module 06

Partha Pratim
Das

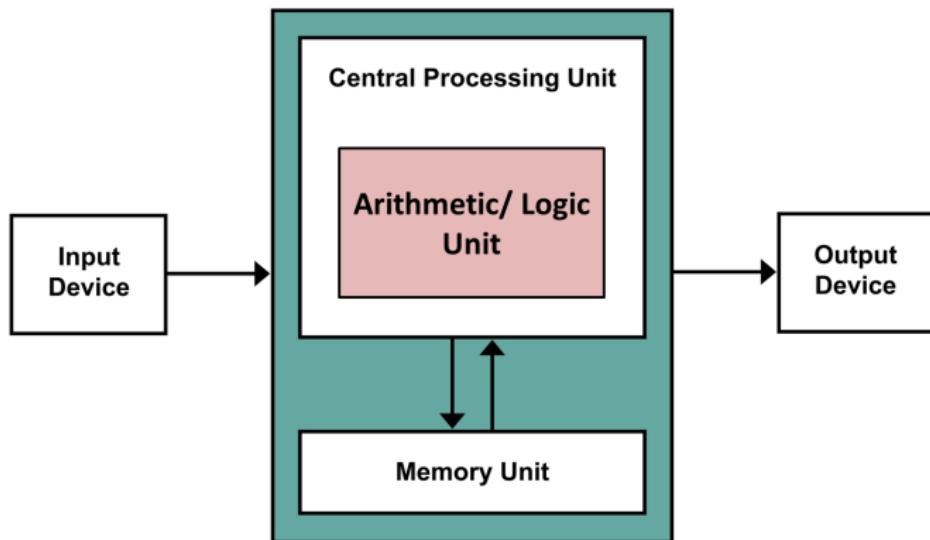
Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary





Generations of Programming Languages

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- First-generation Languages (1954–1958)
- Second-generation Languages (1959–1961)
- Third-generation Languages (1962–1970)
- Fourth-generation Languages (1970–1980)
- Object-orientation Boom (1980–1990)
- Emergence of Frameworks (1990–today)



First-generation Languages (1954–1958)

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- *Major Features*
 - **Formula Translation**
 - **Mathematical Computation**
 - **Notions of Programming:**
 - Variable and Literal
 - Expressions
 - Unconditional & Conditional Flow
- *Languages*
 - FORTRAN I (Mathematical expressions)
 - ALGOL 58 (Mathematical expressions)
 - Flowmatic (Mathematical expressions)
 - IPL V (Mathematical expressions)



Topology of First-generation Languages

Module 06

Partha Pratim
Das

Objectives &
Outline

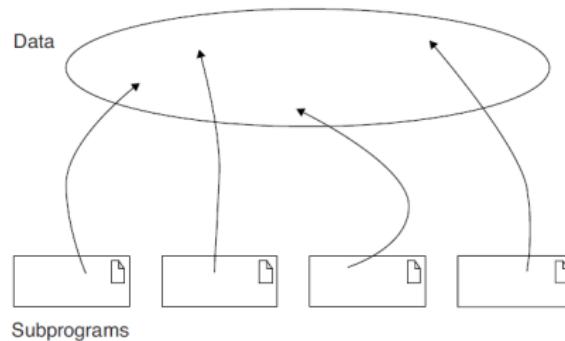
Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Applications in these languages exhibit:
 - Flat physical structure
 - Only of global data and subprograms
 - Error in one part of a program can have effect across the rest of the system



Topology of 1st- and Early 2nd-Generation Languages

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



Second-generation Languages (1959–1961)

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- *Major Features*

- **Blocks and Subprograms**
- **Notions of Programming:**
 - Data Type & Description
 - Statements
 - Parameter Passing Mechanisms
 - Files
 - Pointers
 - List
 - Data Structures

- *Languages*

- FORTRAN II (Subroutines, separate compilation)
- ALGOL 60 (Block structure, data types)
- COBOL (Data description, file handling)
- Lisp (List processing, pointers, garbage collection)



Topology of Second-generation Languages

Module 06

Partha Pratim
Das

Objectives &
Outline

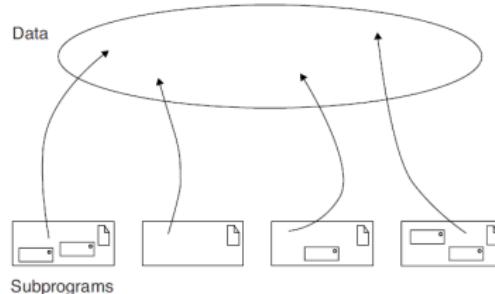
Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Applications in these languages exhibit:
 - Foundations of structured programming - nested subprograms, control structures and scope and visibility of declarations
 - Support parameter-passing mechanisms
 - Fails to address the problems of programming-in-the-large and data design



Topology of Late 2nd- and Early 3rd-Generation Languages

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



Third-generation Languages (1962–1970)

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- *Major Features*

- **Structured Programming**
- **Modularity**
- **Abstraction**
- **Notions of Programming:**
 - Control Constructs
 - Class
- **One-language-fits-it-all**

- *Languages*

- PL/1 (FORTRAN + ALGOL + COBOL)
- ALGOL 68 (Rigorous successor to ALGOL 60)
- Pascal (Simple successor to ALGOL 60)
- Simula (Classes, data abstraction)



Topology of Third-generation Languages

Module 06

Partha Pratim
Das

Objectives &
Outline

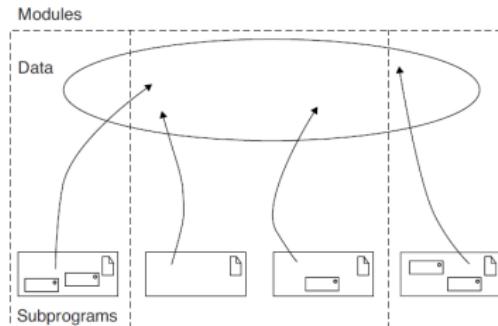
Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Applications in these languages exhibit:
 - Support separately compiled module (group subprograms that were most likely to change together)
 - Dismal support for data abstraction and strong typing, hence such errors could be detected only during execution of the program



Topology of Late 3rd-Generation Languages

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



Fourth-generation Languages (1970–1980)

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- *Major Features*

- **Relational Models for large data handling**
- **Notions of Programming:**
 - Low-level access for Systems Programming
 - Efficiency
- **Moving away from "One-language-fits-it-all"**

- *Languages*

- C (Efficient, small executable)
- FORTRAN 77 (ANSI standardization)
- SQL (Structured Query Language)



Object-orientation Boom (1980–1990)

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- *Major Features*

- **Object Models**
- **Notions of Programming:**

- Strong Typing
- Safety
- Exception

- **Task-specific Languages**

- *Languages*

- Smalltalk 80 (Pure object-oriented language)
- C++ (Derived from C and Simula)
- Ada83 (Strong typing; heavy Pascal influence)
- Eiffel (Derived from Ada and Simula)



Topology of OB and OOP Languages

Module 06

Partha Pratim
Das

Objectives &
Outline

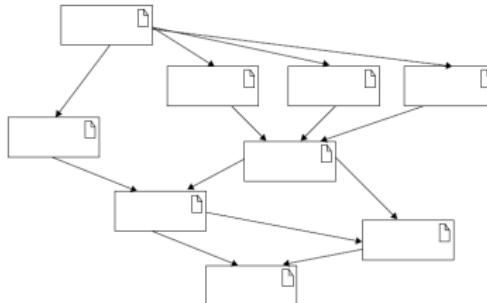
Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Applications in Object Based (OB) and Object Oriented Programming (OOP) languages exhibit:
 - Fundamental logical building blocks are no longer algorithms, but instead are classes and objects
 - Little or no global data
 - Classes, objects, and modules provide an essential yet insufficient means of abstraction



Topology of Small to Medium Applications using OB and OOP Languages

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



Emergence of Frameworks (1990–today)

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

• *Major Features*

- **Programming Frameworks**
- **Use of OOAD for Large Scale Systems**
- **Notions of Programming:**
 - Dynamic Typing
 - Portability
 - Threads
- **Frameworks:**
 - J2SE, J2EE, J2ME (Java-based frameworks for standard, enterprise & mobile computing)
 - .NET (Microsoft's object-based framework)

• *Languages*

- Visual Basic (Development GUI for Windows applications)
- Java (Successor to Oak; designed for portability)
- Python (OO scripting, portable, dynamically typed)
- Visual C# (Java competitor for .NET Framework)
- VB.NET (Visual Basic for .NET Framework)



Topology of Frameworks

Module 06

Partha Pratim
Das

Objectives &
Outline

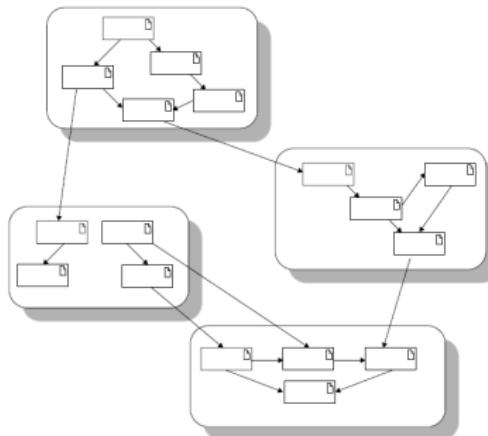
Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Applications in these language frameworks exhibit:
 - Programming-in-the-large
 - At any given level of abstraction, meaningful collections of objects achieve some higher-level behavior



Topology of Large Applications using OOP Language Frameworks

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)
NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das

18



Module Summary

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

- Revisit of the Computation Model
- Study of Generations of Programming Languages
- Understanding of Evolution of Programming



Instructor and TAs

Module 06

Partha Pratim
Das

Objectives &
Outline

Computation

Generations of
Programming
Languages

First
Second
Third
Fourth
OOP
Frameworks

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models
OOA
OOD
OOP

Inter-
relationships

Summary

Module 07: Object-Oriented Analysis & Design

Foundations of the Object Model - OOA, OOD, and OOP

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- Understand the Foundations of Object Models
- Get introduced to the notions of
 - Object Oriented Analysis (OOA),
 - Object Oriented Design (OOD), and
 - Object Oriented Programming (OOP)



Module Outline

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- Foundations of Object Models
 - Object Oriented Analysis (OOA)
 - Object Oriented Design (OOD)
 - Object Oriented Programming (OOP)
- Interrelationships of OOA, OOD, and OOP



Foundations of Object Models

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- OOAD represents an *Evolutionary Development* – not a *Revolutionary* one
- With *Algorithmic Decomposition* alone, only limited amount of complexity can be handled – hence we turn to *Object-Oriented Decomposition*
- The following events have contributed to the evolution:
 - Advances in computer architecture
 - Advances in programming languages
 - Advances in programming methodology
 - Advances in database models
 - Research in artificial intelligence
 - Advances in philosophy and cognitive science



Object Oriented Analysis (OOA)

Module 07

Partha Pratim
Das

Objectives &
Outline

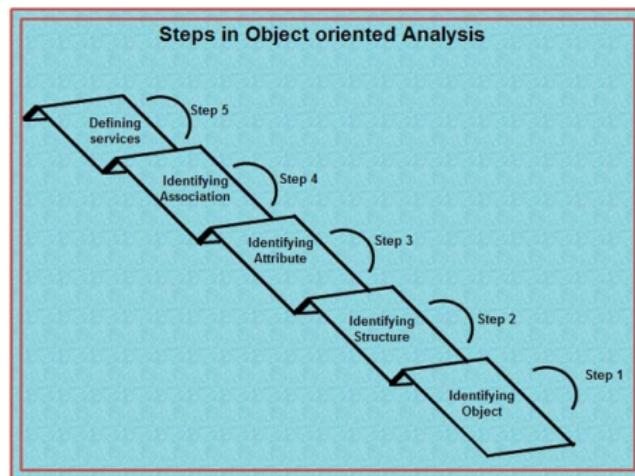
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- **Object-Oriented Analysis** is a method of analysis that examines *requirements* from the perspective of the *classes* and *objects* found in the *vocabulary of the problem domain*





Object Oriented Analysis (OOA)

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- The stages for Object-Oriented Analysis are:

- Identifying Objects
- Identifying Structure
- Identifying Attributes
- Identifying Associations
- Defining Services



Object Oriented Design (OOD)

Module 07

Partha Pratim
Das

Objectives &
Outline

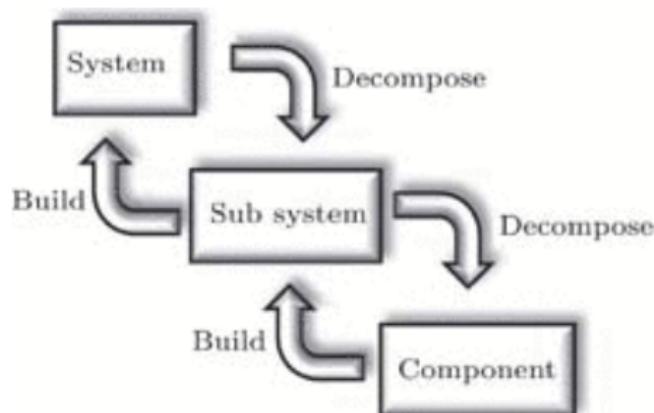
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- **Object-Oriented Design** is designing with:
 - *Object-oriented Decomposition* and
 - Different notations to express different models:
 - *Logical Models* (class and object structure)
 - *Physical Models* (module and process architecture)
 - *Behavioral Models* (statics and dynamics of the system)



Object Oriented Decomposition



Object Oriented Design (OOD)

Module 07

Partha Pratim
Das

Objectives &
Outline

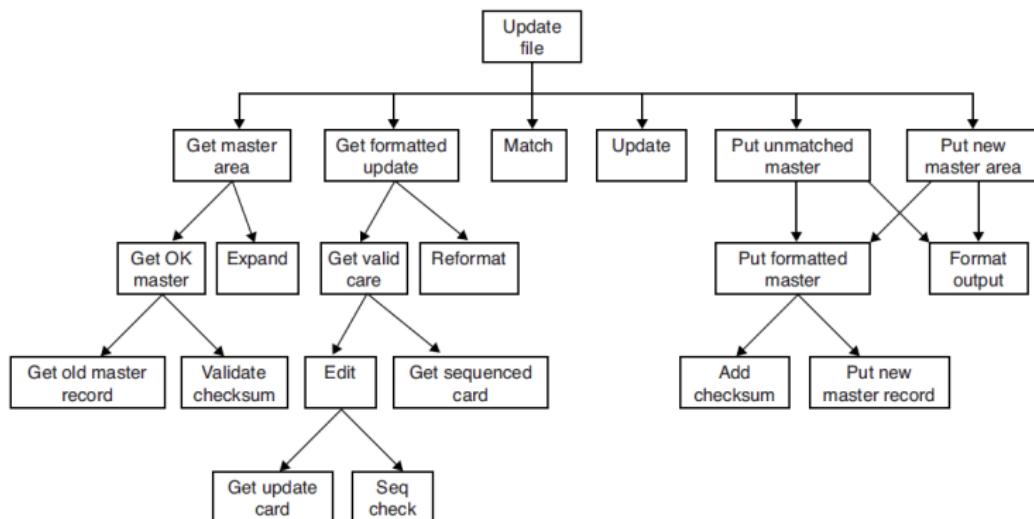
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- Design a program that updates the content of a master file



Algorithmic Decomposition

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Object Oriented Design (OOD)

Module 07

Partha Pratim
Das

Objectives &
Outline

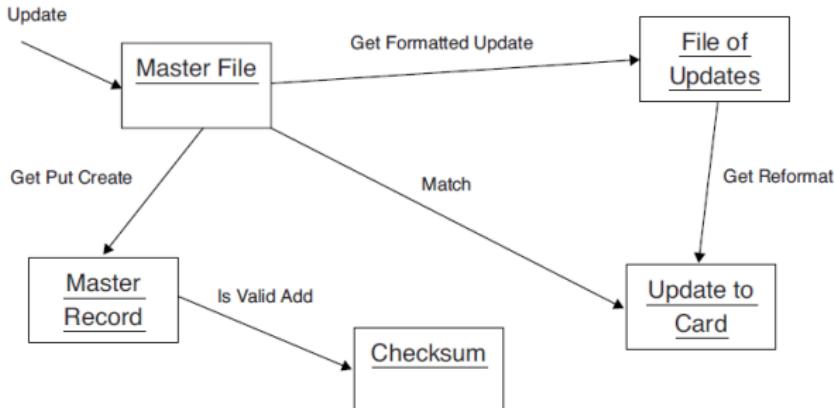
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- Decompose by *key abstractions* in the problem domain
- Identify objects such as *Master File* and *Checksum*, in place of *Get formatted update* and *Add checksum* tasks



Object-Oriented Decomposition

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Object Oriented Design (OOD)

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- Notations to express different models



Unified Modeling Language



Object Oriented Design (OOD)

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- The stages for Object-Oriented Design are:
 - Definition of the context of the system
 - Designing system architecture
 - Identification of the objects in the system
 - Construction of design models
 - Specification of object interfaces
- Object design includes the following phases:
 - Object identification
 - Object representation or the construction of design models
 - Classification of operations
 - Algorithm design
 - Design of relationships
 - Implementation of control for external interactions
 - Package classes and associations into modules



Object Oriented Programming (OOP)

Module 07

Partha Pratim
Das

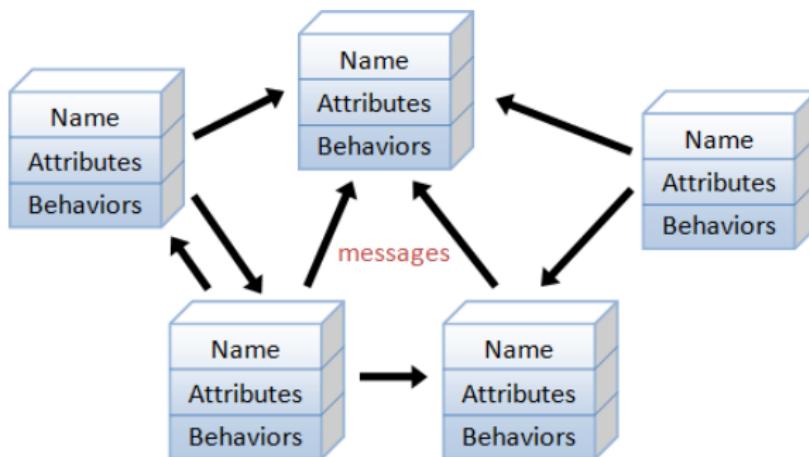
Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary



Objects exchanging messages for Computation



Object Oriented Programming (OOP)

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

There are three important parts of OOP:

- Object
- Class [Type]
- Inheritance



Object Oriented Programming (OOP)

Module 07

Partha Pratim
Das

Objectives &
Outline

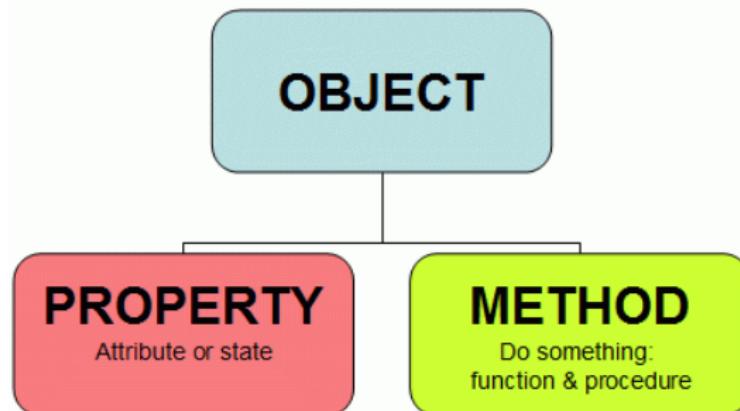
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- ① *Objects* – data abstraction with and interface of named operation and hidden state



Object–Property–Method



Object Oriented Programming (OOP)

Module 07

Partha Pratim
Das

Objectives &
Outline

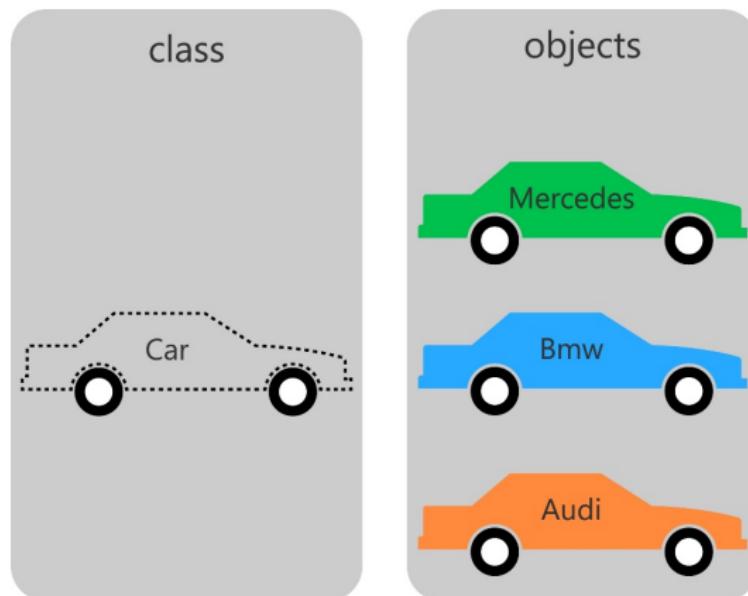
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- ② Objects have an associated *type [class]*



Class [type]–Object



Object Oriented Programming (OOP)

Module 07

Partha Pratim
Das

Objectives &
Outline

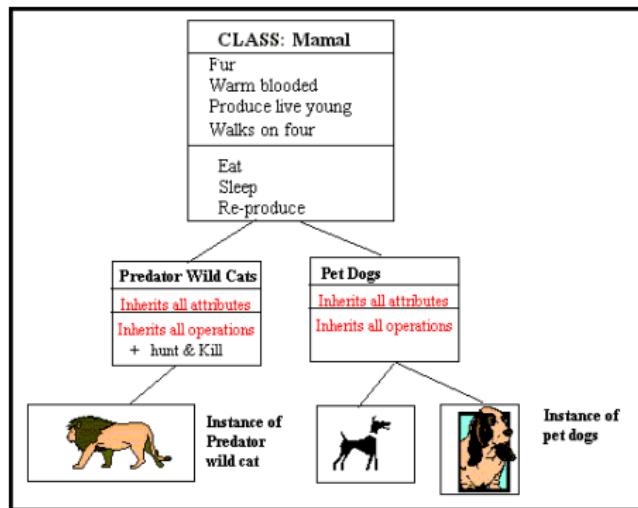
Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- ③ Types [classes] may *inherit* attributes from supertypes [superclasses]



- If a language does not provide direct support for inheritance, then it object-based rather than object-oriented



How are OOA, OOD, and OOP related?

Module 07

Partha Pratim
Das

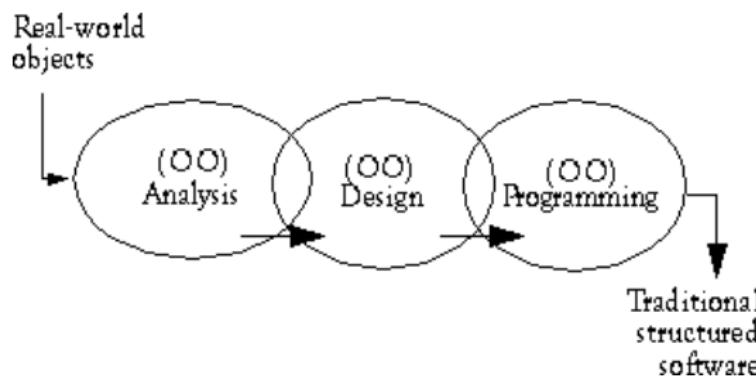
Objectives &
Outline

Foundations
of Object
Models
OOA
OOD
OOP

Inter-
relationships

Summary

- **OOA** is concerned with developing a object model that capture the requirement
- **OOD** is concerned with translating OOA model into a specific model that can be implemented by a software
- **OOP** is concerned with realizing the OOD model using OOP language such as Java or C++





Module Summary

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

- Understand the event contributed in the foundation of the object model
- Understand OOP, OOD and OOA
- Understand how OOA, OOD, and OOP are inter-related



Instructor and TAs

Module 07

Partha Pratim
Das

Objectives &
Outline

Foundations
of Object
Models

OOA
OOD
OOP

Inter-
relationships

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Module 08: Object-Oriented Analysis & Design

Elements of Object Model (Major): Abstraction and Encapsulation

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- Understand the major elements / essential characteristics of Object Models
- Understand *Abstraction*
- Understand *Encapsulation*



Module Outline

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- Elements of Object Model
 - Major Elements
 - Minor Elements
- Abstraction
 - Abstraction Hierarchy
- Encapsulation
- Abstraction vis-à-vis Encapsulation



Elements of Object Model: Major

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model
Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

There are **four major** (*mandatory and essential*) elements of the object model:

- ① *Abstraction*
- ② *Encapsulation*
- ③ *Modularity*
- ④ *Hierarchy*



Elements of Object Model: Minor

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model
Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

In addition, there are **three minor** (*useful but not essential*) elements of the object model:

- ① Typing
- ② Concurrency
- ③ Persistence



Abstraction

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- **Abstraction** is to *simplify a problem by omitting unnecessary details*
 - Focus attention on only one aspect of the problem and ignore irrelevant details
- Also called: *Model Building*
- Example: *Car Brakes*
 - Pressing the pedals will stop the vehicle but you do not need to know how it works!



Abstraction

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

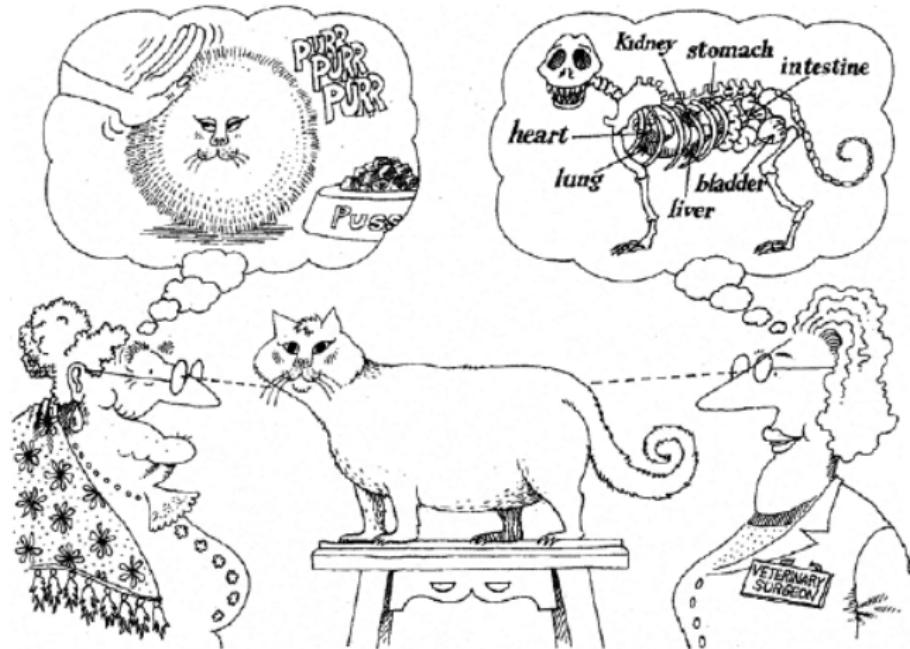
Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary



Essential characteristics of an object is relative to the perspective of the viewer

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)
NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das



Abstraction – Electrical Circuits

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- You are asked to design a circuit
- Would you: Try to compute all possible inputs for all possible outputs, try to characterize all possible states?
- You would possibly focus on various types of models for the circuit



Abstraction – Electrical Circuits

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

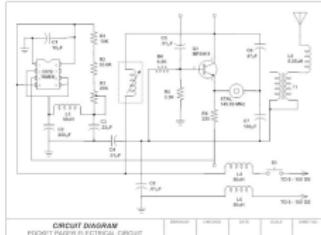
Abstraction

Abstraction
Hierarchy

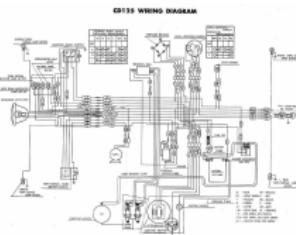
Encapsulation

Abstraction
vis-à-vis
Encapsulation

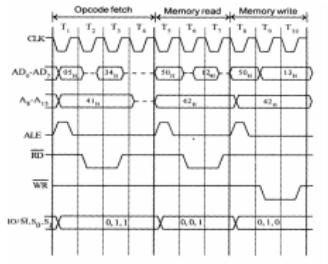
Summary



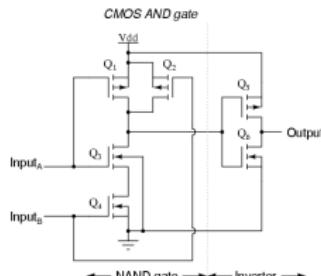
Schematic Diagram



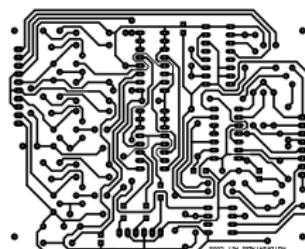
Wiring Diagram



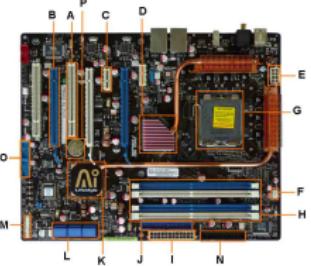
Timing Diagram



CMOS Diagram



Layout Diagram



Motherboard Diagram



Abstraction – Electrical Circuits

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Diagram	Abstraction
Schematic Diagram	– How to analyze electrical behavior?
Wiring Diagram	– How to lay wires to connect elements?
Timing Diagram	– How does the signal change with time?
CMOS Diagram	– How CMOS gates can realize the circuit?
Layout Diagram	– How to interconnect on a PCB?
Motherboard Diagram	– How to pack the components on a PCB?



Abstraction – India

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- You are asked to develop an overall understanding of India
- Would you: Meet all the citizens of India, visit every house, and examine every tree of India?
- You would possibly only refer to various types of maps of India



Abstraction – India

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

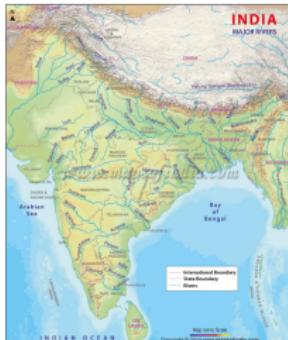
Encapsulation

Abstraction
vis-à-vis
Encapsulation

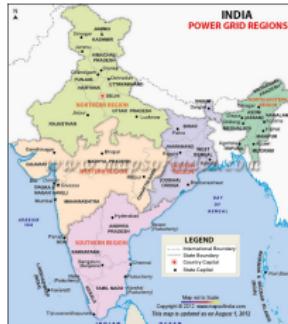
Summary



Political



Rivers



Power-Grid



Vegetation



Railways



World Heritage Site



Abstraction – India

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Map	Abstraction
Political Map	– What is the administrative structure?
Rivers Map	– What is natural distribution for water?
Power-Grid Map	– What is the power distribution profile?
Vegetation Map	– What is the cultivation pattern?
Railways Map	– What is the spread of connectivity?
World Heritage Site Map	– What is the cultural heritage?



Abstraction:

Example – *Hydroponics Gardening System*

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Hydroponics farm:

- Plants are grown in a nutrient solution, without sand, gravel, or other soils
- Control diverse factors such as temperature, humidity, light, pH, and nutrient concentrations
- Design an automated system that constantly monitors and adjusts these elements
- An automated gardener should efficiently carry out growing plans for the healthy production of multiple crops

Key Abstractions:

- Sensors
- Heaters
- Growing Plans



Multiple (Abstract) Views of a Personal Computer:

Google Image Search – *Structure of a Personal Computer*

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

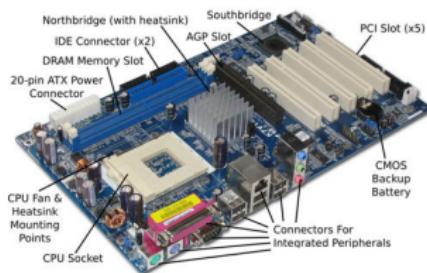
Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

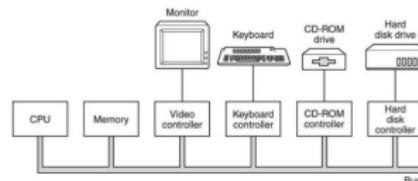


<1> Physical View



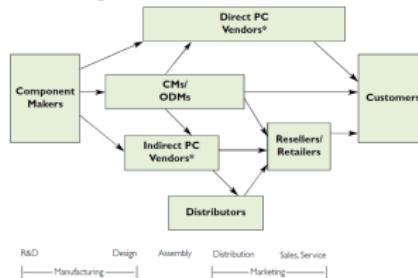
<3> Board View

- 1> http://www.assis.pro.br/public_html/davereed/01-Basics.html
- 2> <http://www.slideshare.net/LiEdo/computer-systems-organization>
- 3> http://ultimatecomputerexperts.blogspot.in/2011/06/blog-post_1232.html
- 4> <http://cnfolio.com/ELMnotes20>



Logical structure of a simple personal computer.

<2> Logical View



<4> Sales View



Abstraction Hierarchy

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction
Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- Several abstractions of the same problem are possible:
 - Focus on **some specific aspect** and **ignore the rest**
 - Use **different models** for **different aspects** of the problem
- For complex problems:
 - A single level of abstraction is inadequate
 - A hierarchy of abstractions needs to be constructed
- Hierarchy of models:
 - A model in a layer is abstraction of the lower layer models
 - Higher layer models implements the model in the layer



Abstraction Hierarchy: Example – *Living Organisms*

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

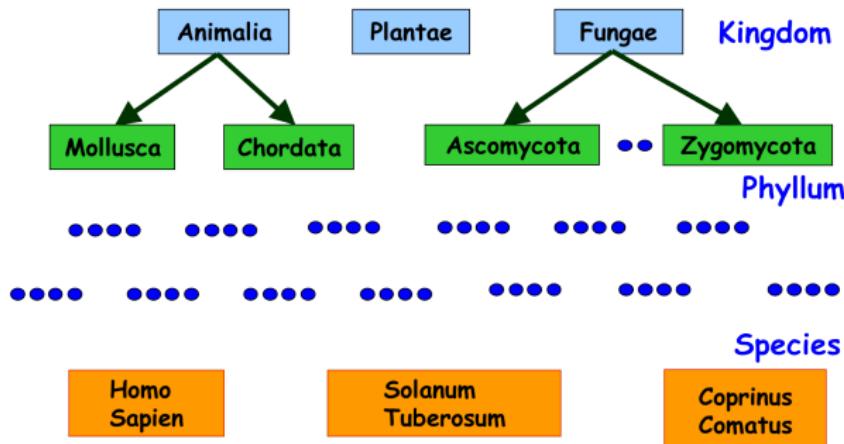
Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- To understand the life forms that inhabit the earth
- If we start examining each living organism:
 - We shall almost never complete it
 - Also, get thoroughly confused
- But we can build an abstraction hierarchy





Abstraction Hierarchy: Example – Employee in LMS

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Abstraction: Employee

Characteristics:

- Name
- ID
- DoB

Responsibilities:

- Record Daily Attendance
- Request for Leave
- Cancel an Approved Leave
- Avail Leave, if approved

IS-A

Abstraction: Executive

Characteristics:

- Reporting_Lead
- ...

Additional Responsibilities:

- Report to Lead

Abstraction: Lead

Characteristics:

- Reporting_Manager
- List of Reporting Executives

Additional Responsibilities:

- Approve Leave (Executive)
- Regret Leave (Executive)
- Revoke Leave (Executive)
- Report to Manager
- Take Reporting (Executive)

Abstraction: Manager

Characteristics:

- List of Reporting Leads
- ...

Additional Responsibilities:

- Approve Leave (Lead)
- Regret Leave (Lead)
- Revoke Leave (Lead)
- Take Reporting (Lead)

- For brevity, all characteristics and all responsibilities are not shown
- Suggest refinements to Employee hierarchy



Abstraction Hierarchy: Example – *Hydroponics Gardening System*

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Abstraction: Sensor

Characteristics:

- Location

Responsibilities:

- Calibrate

IS-A

Abstraction: Temperature Sensor

Characteristics:

- Location
- Temperature

Responsibilities:

- Calibrate
- Report current temperature

IS-A

Abstraction: Active Temperature Sensor

Characteristics:

- Location
- Temperature
- Setpoint

Responsibilities:

- Calibrate
- Report current temperature
- Establish setpoint

• Multi-level hierarchy



Encapsulation

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- **Encapsulation** means to *encapsulate*:
 - Put everything into one envelop, and
 - Provide others to use it through a well-planned orifice
- Encapsulation compartmentalizes the elements of an abstraction that constitute:
 - Structure
 - Behavior



Encapsulation

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

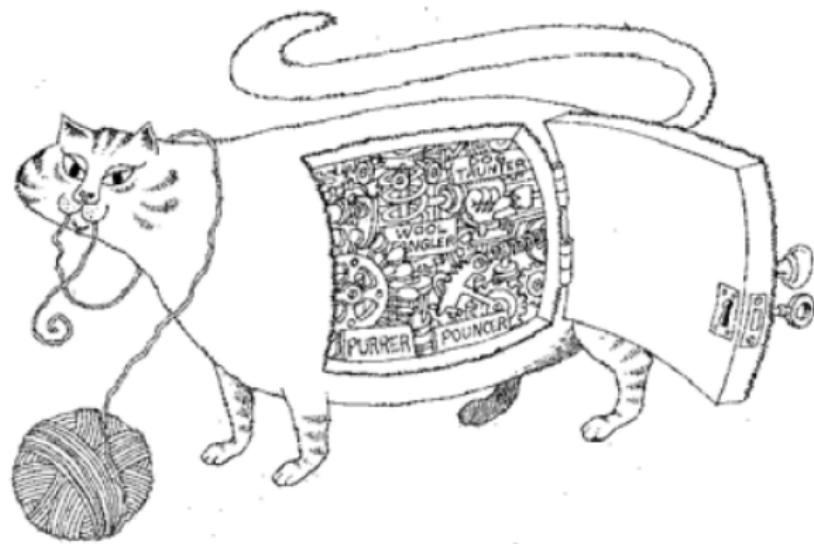
Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary



Encapsulation hides the details of the implementation of an object

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Encapsulation: Example – Personal Computer

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

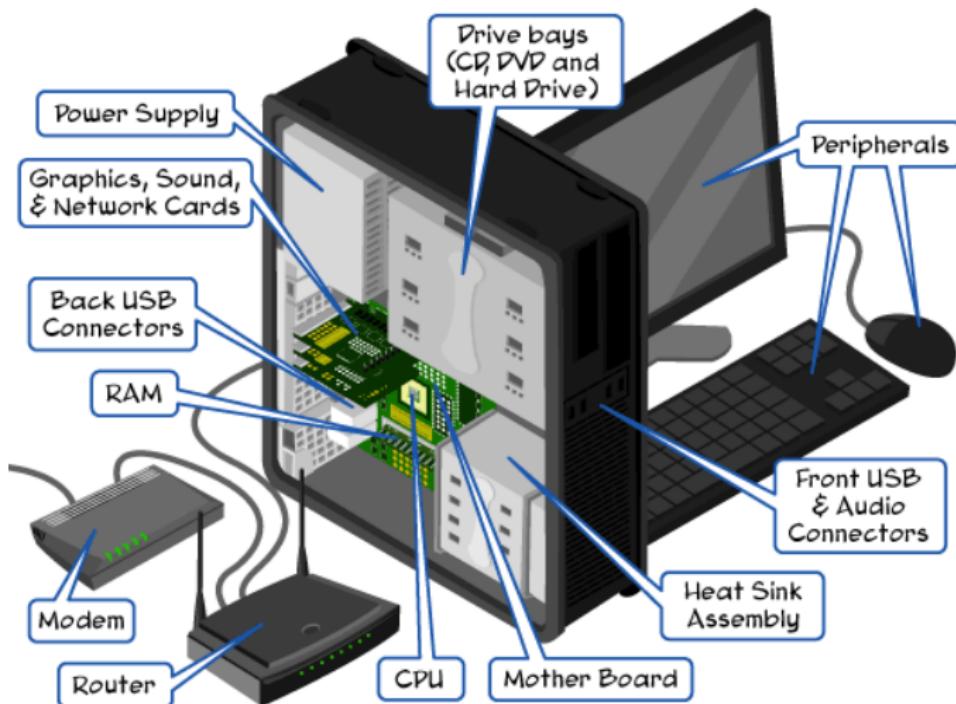
Major
Minor

Abstraction
Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary





Encapsulation: Example – Stack

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- **Structure** is encapsulated in the implementation:

- **store** for the elements:

- Static / Dynamic array
 - Singly-linked list
 - Deque

- **marker** for the **top** element:

- Array index
 - List pointer
 - Encapsulated in Deque

- Hidden from the **external world**

- **Behavior** is exposed through interface:

- **interface** for a **stack**:

- Push
 - Pop
 - Top
 - Empty

- Exposed to the **external world**

Abstraction: Stack

Structure:

- store
- marker

Behavior:

- + Push
- + Pop
- + Top
- + Empty



Encapsulation

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- For an *Abstraction*, **Encapsulation** separates
 - Contractual interface (Push-Pop-Top-Empty) and
 - Implementation (array-list-deque)
- Data encapsulation** ⇒ **Data hiding**

Encapsulation binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse



Encapsulation:

Example – *Employee in LMS*

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Abstraction: Employee

Structure:

- Name
- ID
- DoB

Behavior:

- + Record() // Daily Attendance
- + Request() // Leave
- + Cancel() // An Approved Leave
- + Avail() // Leave, if approved



Encapsulation:

Example – *Hydroponics Gardening System*

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis

Encapsulation

Summary

Abstraction: Heater

Structure:

- Location
- Status

Behavior:

- + Turn_On()
- + Turn_Off()
- + Provide_Status() // If running

Possible Implementations:

- Connect using Memory-mapped I/O
- Connect using Serial Communication Line
- Connect using Wireless

None need to change the interface

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Abstraction vis-à-vis Encapsulation

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Abstraction	Encapsulation
<ul style="list-style-type: none">Solves the problem in the design levelHide the unwanted data and Expose useful dataAllows us to focus on what the object does instead of how it does itProvides Outer layout, used in terms of design	<ul style="list-style-type: none">Solves the problem in the implementation levelHide the code and data into a single unit to protect the data from outside worldHide the internal details or mechanics of how an object does somethingProvides Inner layout, used in terms of Outer

Example

- | | |
|--|---|
| <ul style="list-style-type: none">Look of a Mobile Phone has a display screen and keypad buttons to dial a number | <ul style="list-style-type: none">Implementation detail of a Mobile Phone deals with how keypad button and Display Screen connect with each other using circuits |
|--|---|



Module Summary

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

- Every Object Model needs to be explored in terms of 4 major and 3 minor elements
- *Abstraction* and *Encapsulation* are two major elements that deal with the outer (design) and inner (implementation) aspects of an object respectively
- We study the other two major elements – *Modularity* and *Hierarchy* in the next module



Instructor and TAs

Module 08

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Major
Minor

Abstraction

Abstraction
Hierarchy

Encapsulation

Abstraction
vis-à-vis
Encapsulation

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

Module 09: Object-Oriented Analysis & Design

Elements of the Object Model (Major): Modularity and Hierarchy

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges
Intelligent
Modularization

Hierarchy
Types of
Inheritance

Summary

- Understand the major elements / essential characteristics of Object Models
- Understand *Modularity*
- Understand *Hierarchy*



Module Outline

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Elements of Object Model – Recap
- Modularity
 - Objectives & Challenges
 - Intelligent Modularization
- Hierarchy
 - Types of Inheritance



Elements of Object Model – Recap

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy
Types of
Inheritance

Summary

There are **four major** (*mandatory and essential*) elements of the object model:

- ① Abstraction
- ② Encapsulation
- ③ *Modularity*
- ④ *Hierarchy*

In addition, there are **three minor** (*useful but not essential*) elements of the object model:

- ① Typing
- ② Concurrency
- ③ Persistence



Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

Objectives &
Challenges

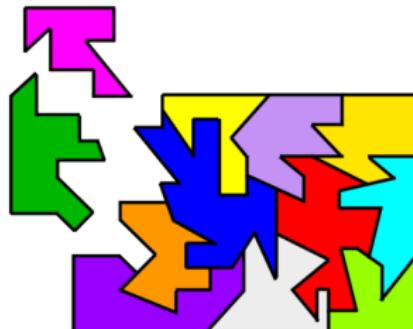
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- **Modularity** is to *partition a program into individual components to manage complexity*
- Modules
 - Compile separately or together
 - Connected with the other module





Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



Modularity packages abstractions into discrete units

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

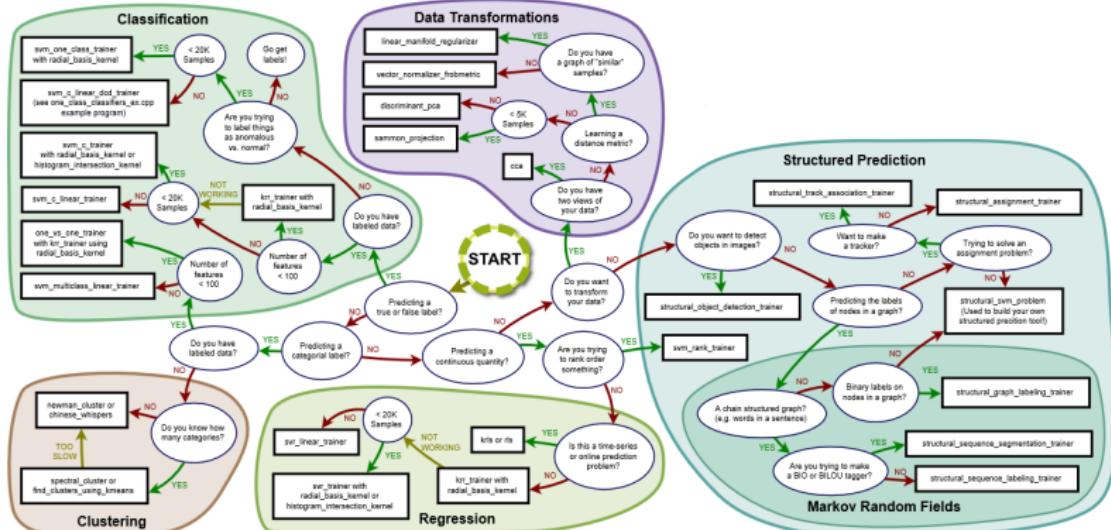
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



Modular decomposition of Machine Learning System



Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

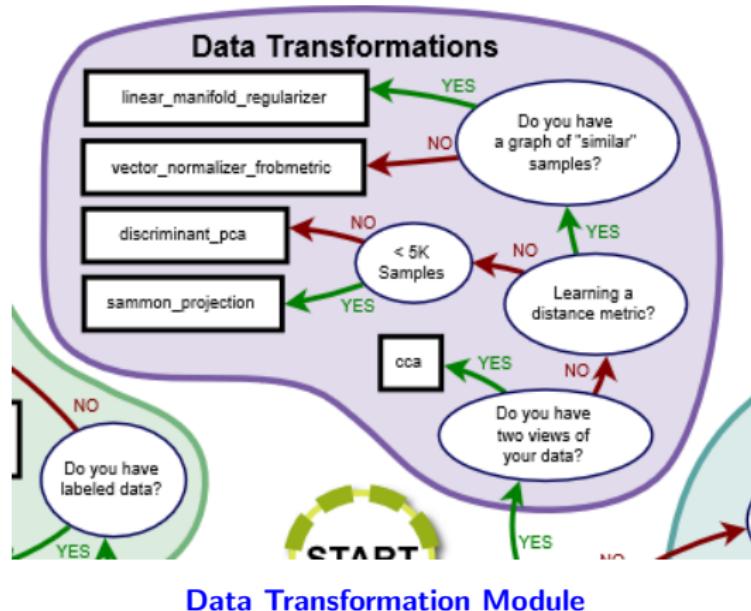
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary





Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

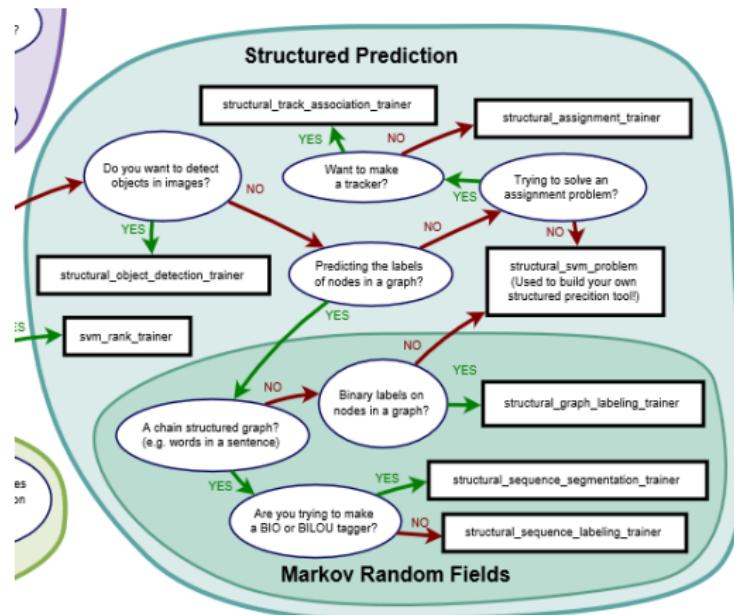
Modularity

Objectives &
Challenges
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



Structured Prediction Module



Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

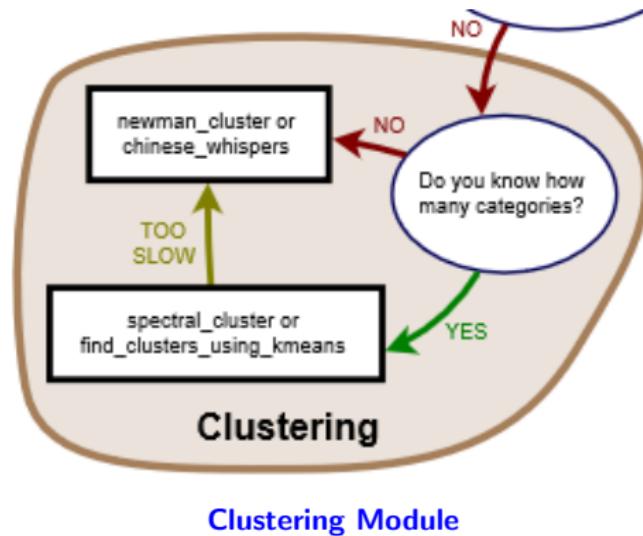
Modularity

Objectives &
Challenges
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary





Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

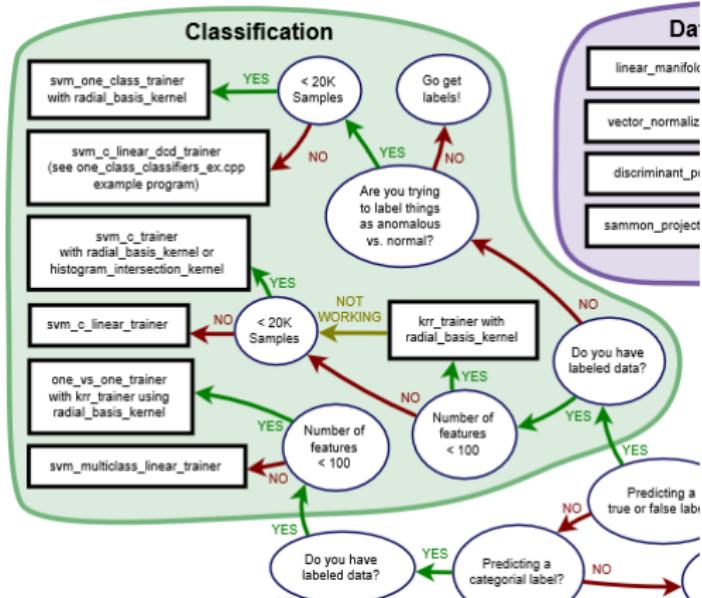
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



Classification Module



Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

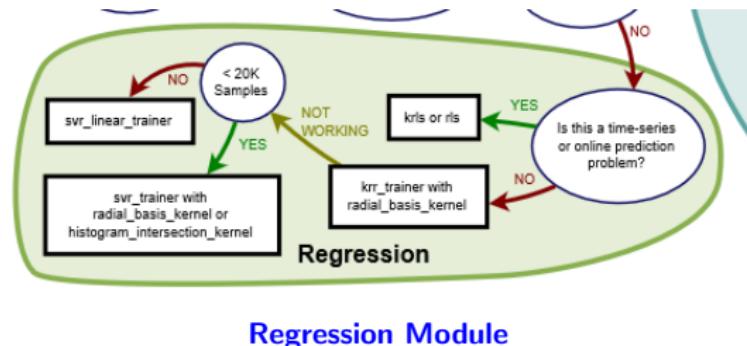
Modularity

Objectives &
Challenges
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary





Example of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

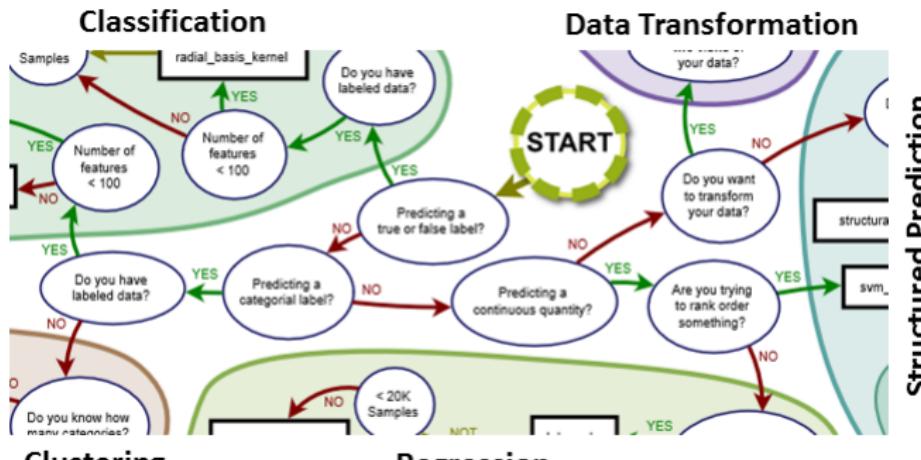
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



Structured Prediction



Objectives of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Modular Decomposition
 - Systematic mechanism to divide problem into individual components
- Modular Composability
 - Enable reuse of existing components
- Modular Understandability
 - Understand module as a unit
 - Easy to change
- Modular Continuity
 - Small changes in system requirements result in changes in individual module rather the whole system
 - Reduces Side-effects
- Modular Protection
 - Errors are localized and do not spread to other modules



Challenges of Modularity

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Decomposition into smaller module may be difficult
- Arbitrary modularization is sometimes worse than no modularization at all

Modularization should follow the semantic grouping of common and interrelated functionality of the system



Challenges of Modularity: Example

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Consider a Distributed System:
 - Runs on a distributed set of processors
 - Uses a message passing mechanism to coordinate the activities of different programs
 - Has several hundred or even a few thousand kinds of messages
- What should be the strategy to modularize?
 - Naive strategy might be to define each message class in its own module
 - This is a singularly poor design decision
 - Creates a documentation nightmare
 - Makes it terribly difficult for any users to find the classes
 - Furthermore, when decisions change, hundreds of modules must be modified or recompiled
 - Shows how information hiding can backfire



Intelligent Modularization (of classes and objects)

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- **Group** logically related abstraction
- **Minimize** dependency among modules
- **Simple** enough to understand fully
- **Ease of Change**
 - Possible to change the implementation of one modules without knowledge of the implementation of other modules and without affecting the behavior of other modules
 - The ease of making a change in the design should bear a reasonable relationship to the likelihood of the change being needed



Modularization:

Example – *Hydroponics Gardening System*

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Suppose we decide to use a commercially available workstation where the user can control the system's operation
- At this workstation, an operator could create new growing plans, modify old ones, and follow the progress of currently active ones
- Key Abstraction
 - Growing plan
 - User Interface



Modularization:

Example – *Hydroponics Gardening System*

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Growing plan

- Create a module whose purpose is to collect all of the classes associated with individual growing plans (for example, `FruitGrowingPlan`, `GrainGrowingPlan`)
- The implementations of these `GrowingPlan` classes would appear in the implementation of this module

- User Interface

- Another module whose purpose is to collect all of the code associated with all user interface functions



Hierarchy

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

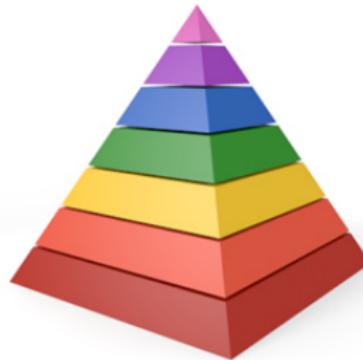
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



(a)



(b)

Source (b): *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Hierarchy

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

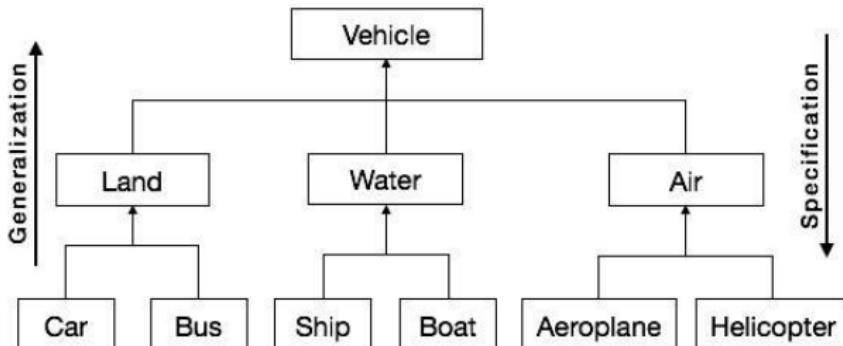
Modularity
Objectives &
Challenges
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Two most important hierarchies in a complex system are
 - Class structure or **is-a** hierarchy (**Abstraction or IS-A**)
 - Object structure or **part-of** hierarchy (**Decomposition or HAS-A**)
- Common structure & behavior are migrated to superclass
 - Superclass represent generalized abstraction
 - Subclasses represent specializations
 - Subclasses can add, modify & hide methods from superclass





Hierarchy

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges
Intelligent
Modularization

Hierarchy
Types of
Inheritance
Summary

- Data abstraction provide an opaque barrier to hide the methods and state
- Inheritance requires to open this interface and allow state & methods to be accessed without abstraction
- Different programming languages trade off support for inheritance in different ways
- C++ and Java offer great flexibility
 - **Private parts** – accessible **only to the class** itself
 - **Protected parts** – accessible **only to the class and its subclasses**
 - **Public parts** – accessible **to all clients**



Single Inheritance

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

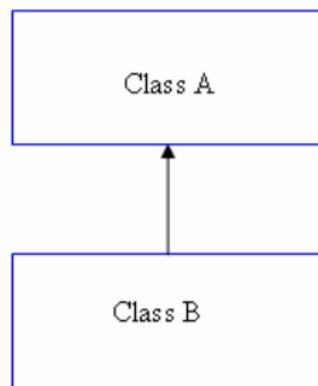
Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary



Hydroponics Gardening System

- FruitGrowingPlan IS-A GrowingPlan
- GrainGrowingPlan IS-A GrowingPlan
- GrowingPlan is general superclass
- Others are specialized subclasses



Single Inheritance

Module 09

Partha Pratim
Das

Objectives &
Outline

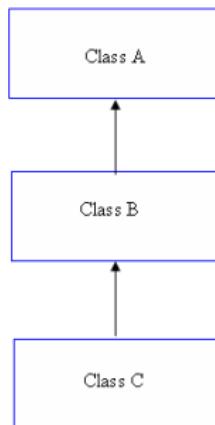
Elements of
Object Model

Modularity
Objectives &
Challenges
Intelligent
Modularization

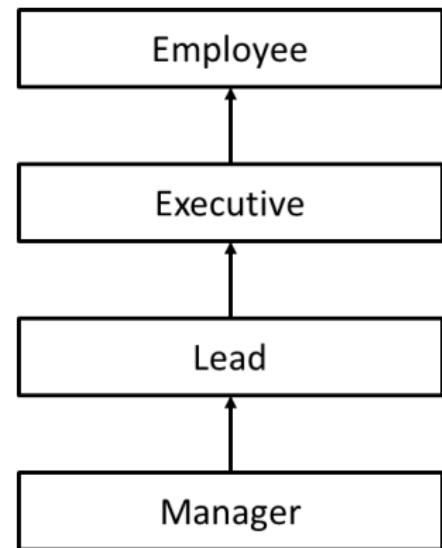
Hierarchy
Types of
Inheritance

Summary

- Multi-level Inheritance



- LMS





Multiple Inheritance

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

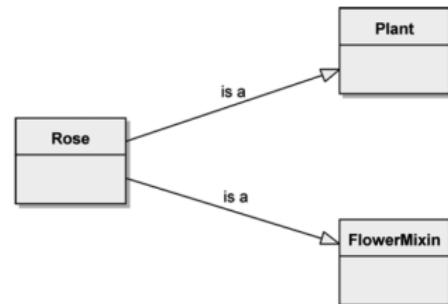
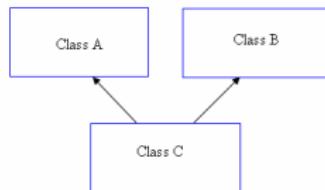
Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- For certain abstractions, it is useful to provide inheritance from multiple superclasses
- Example: The Rose Class Inherits from Multiple Superclasses





Hierarchical Inheritance

Module 09

Partha Pratim
Das

Objectives &
Outline

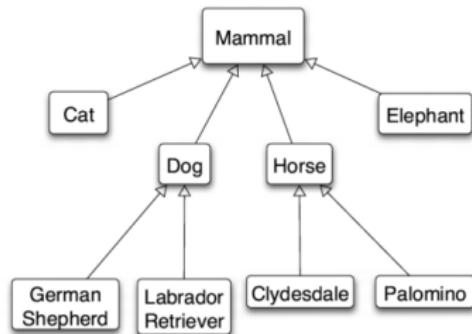
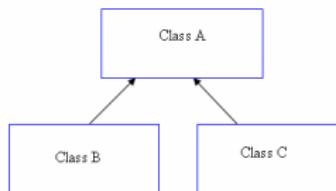
Elements of
Object Model

Modularity
Objectives &
Challenges
Intelligent
Modularization

Hierarchy
Types of
Inheritance

Summary

- More than one sub classes are created from the same super class
- Example: Cat, Dog, Horse and Elephant classes are created from single Superclasses Mammal





Hybrid Inheritance

Module 09

Partha Pratim
Das

Objectives &
Outline

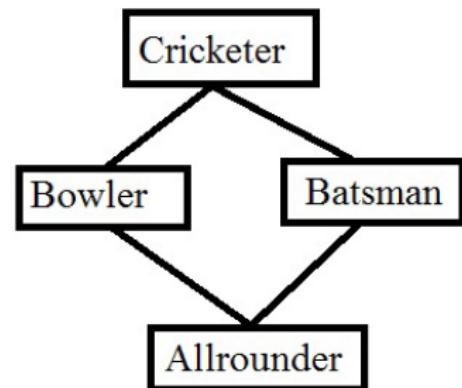
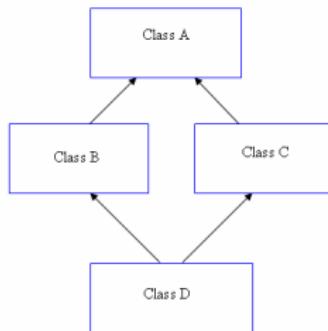
Elements of
Object Model

Modularity
Objectives &
Challenges
Intelligent
Modularization

Hierarchy
Types of
Inheritance

Summary

- It may be a combination of
 - Multilevel and Multiple inheritance or
 - Hierarchical and Multilevel inheritance or
 - Hierarchical and Multiple inheritance





Module Summary

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity
Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

- Every Object Model needs to be explored in terms of 4 major and 3 minor elements
- *Modularity* and *Hierarchy* are two major elements that deal with the separation of concerns in implementation and inheritance aspects of an object respectively
- We study the three minor elements – *Typing*, *Concurrency* and *Persistence* in the next module



Instructor and TAs

Module 09

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Modularity

Objectives &
Challenges

Intelligent
Modularization

Hierarchy

Types of
Inheritance

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary

Module 10: Object-Oriented Analysis & Design

Elements of the Object Model (Minor): Typing, Concurrency, & Persistence

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency

Process &
Thread

Persistence
Continuum

Summary

- Understand the minor elements / useful characteristics of Object Models
- Understand *Typing*
- Understand *Concurrency*
- Understand *Persistence*



Module Outline

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary

- Elements of Object Model – Recap
- Typing
 - Types of Types
 - Polymorphism
- Concurrency
 - Process & Thread
- Persistence
 - Continuum



Elements of Object Model – Recap

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary

There are **four major** (*mandatory and essential*) elements of the object model:

- ① Abstraction
- ② Encapsulation
- ③ Modularity
- ④ Hierarchy

In addition, there are **three minor** (*useful but not essential*) elements of the object model:

- ① *Typing*
- ② *Concurrency*
- ③ *Persistence*



Typing

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum
Summary

- **Typing** is the *enforcement of the class of an object*, such that objects of different types
 - may not be interchanged, or at the most,
 - they may be interchanged only in very restricted ways
- **Type** derives from the theories of *abstract data types*
- A **type** is a precise characterization of *structural or behavioral properties* which a collection of entities all **share**
- We use the terms **type** and **class** *interchangeably*



Typing

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

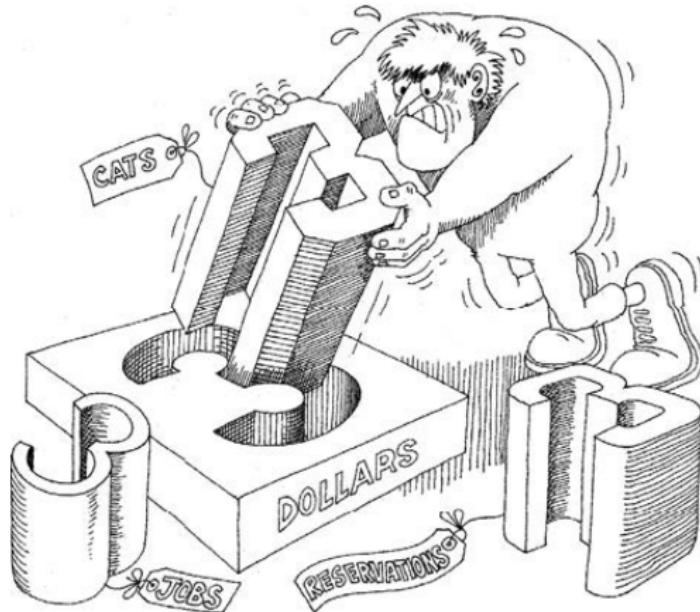
Typing

Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary



Strong typing prevents mixing of abstractions

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



Typing

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary

- Built-in (Primitive) Types

- int
- double
- char
- bool
- ...

- User-Defined Types (UDT)

- Complex
- Vector
- String
- Employee
- Executive
- ...
- Leave
- CasualLeave
- ...



Type of a Programming Language

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary

A Programming Language may be

- *Statically typed* – static binding or early binding

- Types are associated with variables not values [C]

```
int iVal = 2;           // iVal is of type int
double dVal = 3.6;     // dVal is of type double
iVal = 3.6;            // Implicit conversion of double --> int
dVal = 2;              // Implicit conversion of int --> double
```

- Static type checking is the process of verifying the type safety of a program based on analysis of a program's text

- Example: C, C++, Java, Ada

- *Dynamically typed* – dynamic binding or late binding

- Types are associated with values not variables [Python]

```
iVal = 2    // iVal is of type int
dVal = 3.6  // dVal is of type double
iVal = 3.6  // iVal is of type double
dVal = 2    // dVal is of type int
```

- Dynamic type checking is the process of verifying the type safety of a program at run-time

- Example: Python, C++ and Java (polymorphism), Smalltalk



Type of a Programming Language

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary

A Programming Language may be

- *Strongly typed*

- Typing errors are prevented at runtime
- Allow little implicit type conversion
- Does not use static type checking
- Compiler does not check or enforce type constraint rules
- Example: Python, C++ (void* of C), Java

- *Weakly typed*

- Easy to use a value of one type as if it were a value of another type
- Allow implicit type conversion (in a type-safe manner?)
- Example: C, Ada, Some features of C++ and Java

- *Untyped*

- There is no type checking
- Any type conversion required is explicit
- Example: Assembly language, Smalltalk



Polymorphism

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum
Summary

- **Polymorphism** exists when **dynamic typing** and **inheritance** interact
- A single name (such as a variable) may denote objects of many different classes that are related by some common superclass
- **Monomorphism**, in contrast, is supported by languages that are both strongly and statically typed

Polymorphism is the most powerful feature of object-oriented programming languages next to the support for abstraction



Concurrency

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency

Process &
Thread

Persistence
Continuum

Summary

- **Concurrency** is *the property that distinguishes an active object from one that is not active*
- Allows multiple tasks to execute, interact and collaborate at the same time to achieve the global functionality
- Concurrency is critical for *Client-Server Model* of computation



Concurrency

Module 10

Partha Pratim
Das

Objectives &
Outline

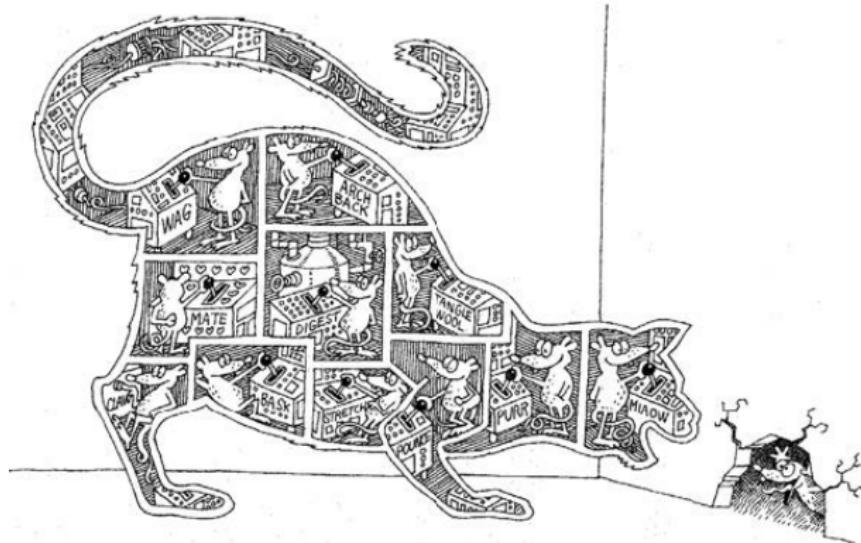
Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum

Summary



Concurrency allows different objects to act at the same time

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Concurrency:

Example – *Mobile Phone*

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum
Summary

In a mobile phone many tasks happen at the same time

- **Scenario 1:** While editing the address book

- Connected to the Base Station
- An SMS is delivered
- Notification raised

- **Scenario 2:** While on a call

- Connected to the Base Station
- An SMS is delivered
- A second call arrives

- **Scenario 3:** While watching a video

- Connected to the Base Station
- An App is downloaded and updated
- A WhatsApp message delivered
- Notification raised



Heavyweight and Lightweight Concurrency

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency

Process &
Thread

Persistence
Continuum

Summary

- **Heavyweight** concurrency

- A **heavyweight process** is typically independently **managed** by the target OS and has **its own address space**
- **Communication** among heavyweight processes is **expensive** and involves **inter-process communication**
- Commonly called a **Process**

- **Lightweight** concurrency

- A **lightweight process** lives within a single OS process along with other lightweight processes and **share the same address space**
- **Communication** among lightweight processes is **less expensive** and often involves **shared data**
- Commonly called a **Thread**



Persistence

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence

Continuum

Summary

- **Persistence** is *the property of an object through which its existence transcends time*

Time Object continues to exist after its creator ceases to exist

Space Object's location moves from the address space in which it was created



Persistence

Module 10

Partha Pratim
Das

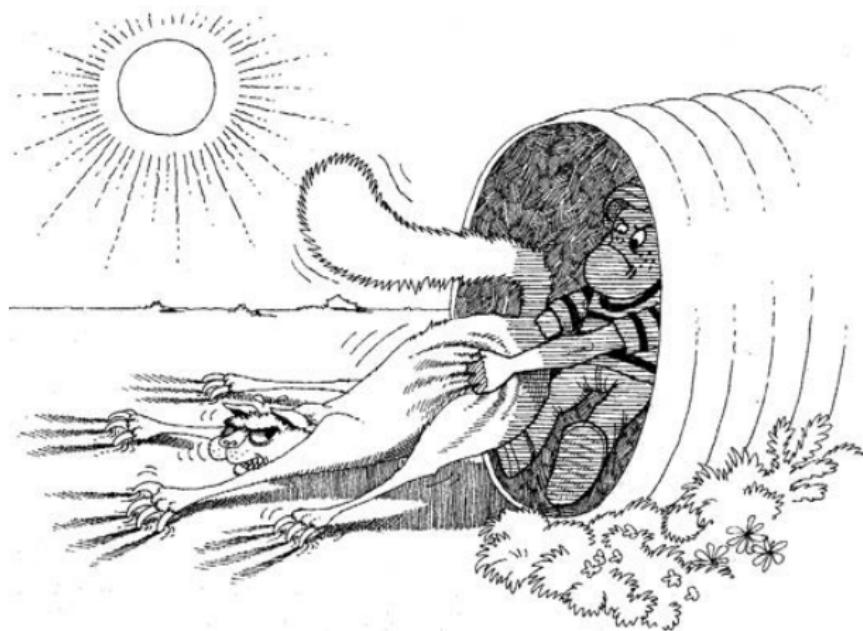
Objectives &
Outline

Elements of
Object Model

Typing
Types of Types
Polymorphism

Concurrency
Process &
Thread

Persistence
Continuum
Summary



Persistence saves the state and class of an object across time or space

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



Continuum of Object Existence

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency

Process &
Thread

Persistence
Continuum

Summary

- An **object** in software takes up some **amount of space** and exists for a particular **amount of time**
- There is a **continuum of object existence**, ranging from **transitory objects** that arise within the evaluation of an expression to **objects in a database** that outlive the execution of a single program
- Spectrum of object persistence encompasses:
 - Transient results in expression evaluation
 - Local variables in function invocation
 - Global variables, and heap items whose extent is different from their scope
 - Data that exists between executions of a program
 - Data that exists between various versions of a program
 - Data that outlives the program



Module Summary

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency

Process &
Thread

Persistence
Continuum

Summary

- Every Object Model needs to be explored in terms of 4 major and 3 minor elements
- *Typing* is graded between strong, weak, and none; can be enforced by static, dynamic, or mixed modes; and is critical for OOP
- *Concurrency* provides the core ability to build client-server system where both can work concurrently
- *Persistence* decides the fate of objects over time and space



Instructor and TAs

Module 10

Partha Pratim
Das

Objectives &
Outline

Elements of
Object Model

Typing

Types of Types
Polymorphism

Concurrency

Process &
Thread

Persistence
Continuum

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object
Summary

Module 11: Object Oriented Analysis & Design

Nature of an Object: State, Behavior, and Identity

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

- Understanding the State, Behavior and Identity of an Object



Module Outline

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object
Summary

- State
 - Validity
 - Constraints
 - Elevator – example
 - Space
- Behavior
 - Common Operations
 - Constraints
 - Roles and Responsibilities
- Identity
 - Display Object – example



What is an Object?

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

- An object must exist
- An object must interact
- An object must be distinguishable



What is not an object?

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

A car has an engine (an object), a steering (an object), a body (an object), ...

- The engine is imported (a property – not an object)
- The steering is driven by power (a property – not an object)
- The color of the body is red (a property – not an object)



What does an Object have?

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

An object is an entity that has STATE, BEHAVIOR, and IDENTITY

The structure and behavior of similar objects are defined in their common class

The terms instance and object are interchangeable



STATE of an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

The **State** of an Object is a combination of values for its properties:

- Consider a Complex number having two properties:

Complex

- re: double // Real Part
- im: double // Imaginary Part

- Its states are possible pairs of values of re and im. For example:

- (2.3, 7.4)
- (-17.3627, 12.9)
- (29.0, -11.11)



STATE of a Date Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

All combination of values of properties of an object may not define valid states:

- Consider a Date having three properties:

Date
– date: {1..31}
– month: {1..12}
– year: {1700..2200} // A 500-year span

- Possible triplets of values of `date`, `month`, and `year` are:

- (20, 7, 2016)
- (1, 1, 1951)
- (15, 8, 1947)

These are **valid dates** and **valid states**

- Several triplets are, however, not valid dates even though the properties individually have valid values:

- (31, 6, 2016) // June does not have 31 days
- (29, 2, 2003) // 2003 is not a leap year
- (12, 9, 1752) // 11 days - 3rd to 13th September 1752 skipped¹

These are **invalid dates** and hence **invalid states**

¹ To change from Julian to Gregorian Calendar



STATE encodes Constraints

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

- Two (or more) objects may have properties of same types; but their states may differ

FirstQuadrant

- x: unsigned int
- y: unsigned int

All states are valid

Fraction

- d: unsigned int
- n: unsigned int

(18, 24) is invalid (reducible) while (3, 4) is valid
(5, 0) is invalid

ProperFraction

- d: unsigned int
- n: unsigned int

(7, 5) is invalid (> 1) while (5, 7) is valid

ChessBoard

- d: unsigned int
- n: unsigned int

(9, 8) is invalid (only 8 X 8 board) while (5, 7) is valid

- All objects have two unsigned int properties; but different valid states
- State succinctly encodes an object (in terms hidden constraints)**



STATE of an Elevator

Module 11

Partha Pratim
Das

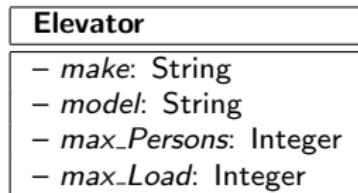
Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

Consider an elevator in the Out-Patient Department (OPD) of a two-storied hospital (moves between Ground Floor and First Floor):



- These **properties**² define a basic elevator
- This model is not interesting because it does not capture the *elevation* (moving up) and *delevation* (moving down) **behavior**³
- So, let us add the behavior to the model

² attributes, characteristics

³ method, responsibility



An Elevator

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object
Summary

Elevator	
- <i>make</i> :	String
- <i>model</i> :	String
- <i>max_Persons</i> :	Integer
- <i>max_Load</i> :	Integer
+ <i>Up()</i>	// Go Up
+ <i>Down()</i>	// Go Down
+ <i>Floor()</i>	// Current Floor
+ <i>Moving()</i>	// IsMoving?
+ <i>Stop()</i>	
+ <i>Open()</i>	// Open Door
+ <i>Close()</i>	// Close Door

- For these we need to track other properties:
 - floor*: {1, 2, X} – current floor decides if *Up()* / *Down()* is valid
 - isMoving*: {Still, Up, Down} – is *Stop()* / *Open()* / *Close()* valid?
 - doorOpen*: Bool – is *Up()* / *Down()* / *Open()* / *Close()* valid?
- Note: *Get()* / *Set()* on attributes are ignored



An Elevator

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object

Summary

Elevator	
- <i>make</i> :	String
- <i>model</i> :	String
- <i>max_Persons</i> :	Integer
- <i>max_Load</i> :	Integer
- <i>floor</i> :	{1, 2, X}
- <i>isMoving</i> :	{Still, Up, Down}
- <i>doorOpen</i> :	Bool
+ <i>Up()</i>	// Go Up
+ <i>Down()</i>	// Go Down
+ <i>Floor()</i>	// Current Floor
+ <i>Moving()</i>	// IsMoving?
+ <i>Stop()</i>	
+ <i>Open()</i>	// Open Door
+ <i>Close()</i>	// Close Door

- **Static** Properties – *does not change* for an instance
- **Dynamic** Values – *changes regularly* for an instance



An Elevator – States

Module 11

Partha Pratim
Das

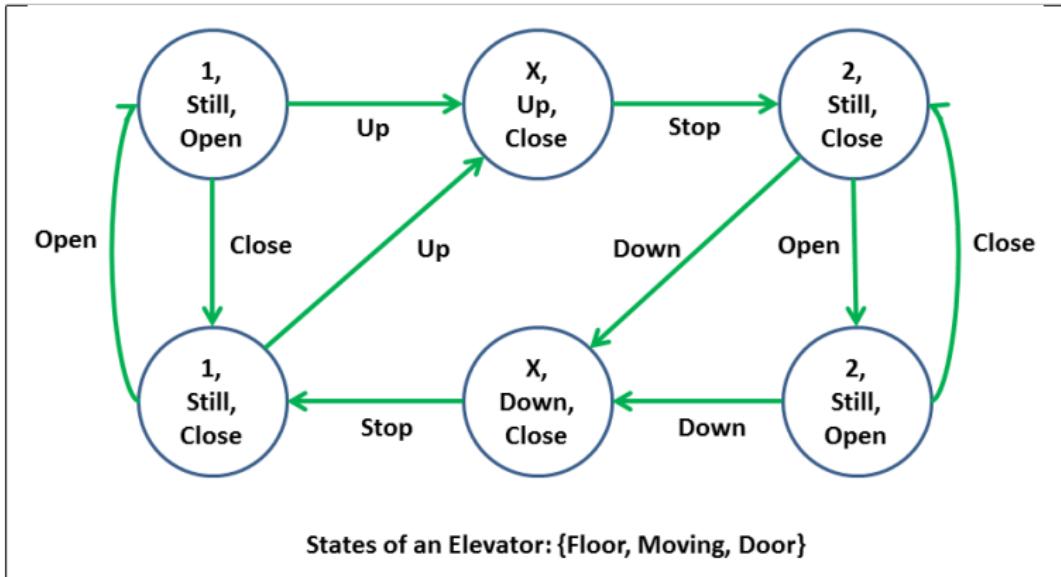
Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object

Summary



The elevator moves through these states as it operates



STATE of an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

The state of an object encompasses all of the (usually static) properties of the object plus the current (usually dynamic) values of each of these properties



Space for an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

All objects within a system encapsulate some state and that all of the state within a system is encapsulated by objects

As every object has a state, hence it takes up some space, it might be in physical world or computer memory



Space for an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

If we analyze the Leave Management System, every Employee is an object, and occupies separate space

Employee	Employee
Name: JOHN EID: E01 Gender: Male OnDuty: True Salary: 20000 DOJ: 14.07.1987	Name: LINDA EID: E05 Gender: Female OnDuty: False Salary: 25000 DOJ: 15.08.1988



BEHAVIOR of an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

The Behavior of an Object is the collection of its **operations**:

- Consider the Complex number objects:

Stack
- store: char[]
- marker: int
+ Push(int): void
+ Pop(): void
+ Top(): char
+ Empty(): bool
+ Print(): void

- It supports 4 common stack operations
- In addition, there will be Constructor, Destructor etc.
- Print() is not a usual stack operation – included for debugging and illustration
- Stack cannot be used to Search() an item!



Client-Server Computing Model – Recap

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

- No object exists in isolation
- Objects are acted on and themselves act on other objects
- Leads to the **Client-Server Model** of computing where
 - Behavior is
 - Services provided by an object
 - Services are requested by
 - Sending Messages, Invoking Operations
 - In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



Common Operations of an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

- **Modifier:** An operation that alters the state of an object – `Push()`, `Pop()`
- **Selector:** An operation that accesses the state of an object but does not alter the state – `Top()`, `Empty()`
- **Iterator:** An operation that permits all parts of an object to be accessed in some well-defined order – `Print()`
- **Constructor:** An operation that creates an object and/or initializes its state
- **Destructor:** An operation that frees the state of an object and/or destroys the object itself



Common Operations for Employee (LMS)

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

- **Modifier:** *set_name(), set_id(), set_salary(), etc in the Employee Object, which updates the name, id and Salary of an Employee*
- **Selector:** *get_name(), get_id(), get_salary(), etc in the Employee Object which reads the name, id and Salary of an Employee*
- **Iterator:** *print_name(), print_id(), print_salary(), etc in the Employee Object, which prints the name, id and Salary of an Employee*
- **Constructor:** *Employee() in the Employee Object, which initializes the object to an initial value.*
- **Destructor:** *~Employee() in the Employee Object*

Employee

- | |
|----------------------------|
| – Name: Shankar |
| – EID: 14C9X7 |
| – Designation: Executive |
| – Gender: male |
| – OnDuty: true |
| – Salary: 1000000.00 |
| – DoJ: 19-Jun-2014 |
| + Employee(...) |
| + ~Employee(...) |
| + get_name(): string |
| + set_name(string): void |
| + get_id(): string |
| + set_id(string): void |
| + get_salary(): double |
| + set_salary(double): void |
| + print_name(): void |
| + print_salary(): void |
| + print_id(): void |



Roles and Responsibilities

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object

Summary

- The behavior of an Object may be logically grouped based on commonality and interdependence. These denote the **Roles** an Object can play.
- Every role is characterized by the services rendered for the role. These are called **Responsibilities**.
- In the LMS, an Employee has the following **Roles**:
 - Executive – primarily has to work for the organization
 - Leadership – take reporting and guide others
 - Approver – approve, regret, revoke leave etc.
 - Management – decide on quantum of leave etc.



Roles and Responsibilities

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

Employee
<ul style="list-style-type: none">- Name: Shankar- EID: 14C9X7- Desgn: Executive- Gender: male- OnDuty: true- Salary: 100000.00- DoJ: 19-Jun-2014 <ul style="list-style-type: none">+ recordAttendance()+ requestLeave()+ cancelLeave()+ availLeave()

Employee
<ul style="list-style-type: none">- Name: Ravi- EID: 08B8Z5- Desgn: Lead- Gender: male- OnDuty: true- Salary: 300000.00- DoJ: 12-May-2008 <ul style="list-style-type: none">+ recordAttendance()+ requestLeave()+ cancelLeave()+ availLeave()+ approveLeave()+ revokeLeave()

Employee
<ul style="list-style-type: none">- Name: Kamala- EID: 02F3W8- Desgn: Manager- Gender: female- OnDuty: true- Salary: 800000.00- DoJ: 05-Aug-2002 <ul style="list-style-type: none">+ recordAttendance()+ requestLeave()+ cancelLeave()+ availLeave()+ approveLeave()+ revokeLeave()+ creditDebitLeave()

● Executive

● Executive
● Leadership
● Approver

● Executive
● Leadership
● Approver
● Management



BEHAVIOR of an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object

Summary

Behavior is how an object acts and reacts, in terms of its state changes and message passing

The state of an object represents the cumulative results of its behavior



IDENTITY of an Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

The **Identity** of an Object that property of an object which distinguishes it from all other objects:

- Objects may be distinguishable by the state if it has at least one unique property
 - EID in Employee helps distinguish
 - Complex objects may not be distinguishable by re & im
- Objects may be distinguishable by an identity ascribed by system:
 - Framework may attach an ID (in Java)
 - Address of the Object may (loosely) alias for an ID (in C++)



IDENTITY of a Display Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

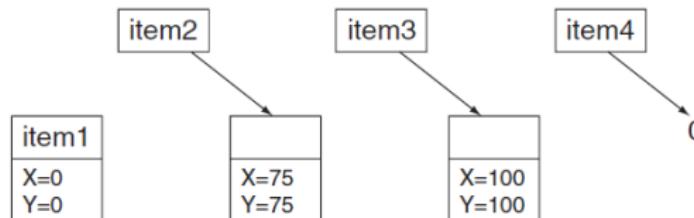
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

A display item is a common abstraction in all GUI-centric systems:

- DisplayItem class is instantiated into item1, item2, item3 and item4
- item1 is the name of a distinct DisplayItem object, item2, item3 actually point to distinct DisplayItem objects. item4 designates no such object



Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



IDENTITY of a Display Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

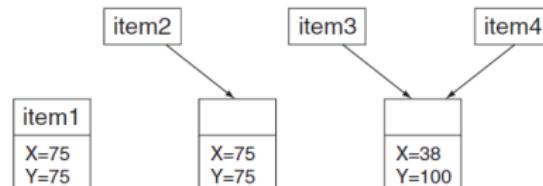
IDENTITY

Display Object

Summary

The unique identity (necessarily the name) of each object is preserved over the lifetime of the object, even when its state is changed

- For example, let us move item1. We can access the object designated by item2, get its location, and move item1 to that same location
- Also, if we equate item4 to item3, we can now reference the object designated by item3 by using item4 also
- Using item4 we can then move that object to a new location, say, X=38, Y=100. DisplayItem object, item2, item3 actually point to distinct



Although item1 and the object designated by item2 have the same state, they represent distinct objects

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



IDENTITY of a Display Object

Module 11

Partha Pratim
Das

Objectives &
Outline

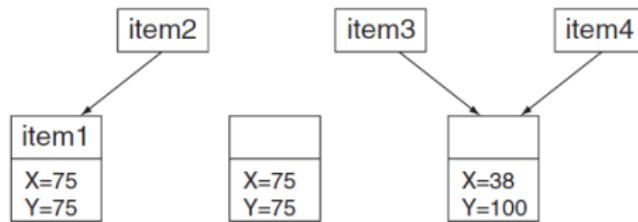
STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

What if we modify the value of the item2 pointer to point to item1?

- Now item2 designates the same object as item1



The object originally designated by item2 can no longer be named, either directly or indirectly, and so its identity is lost

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



A Hammer Object

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE

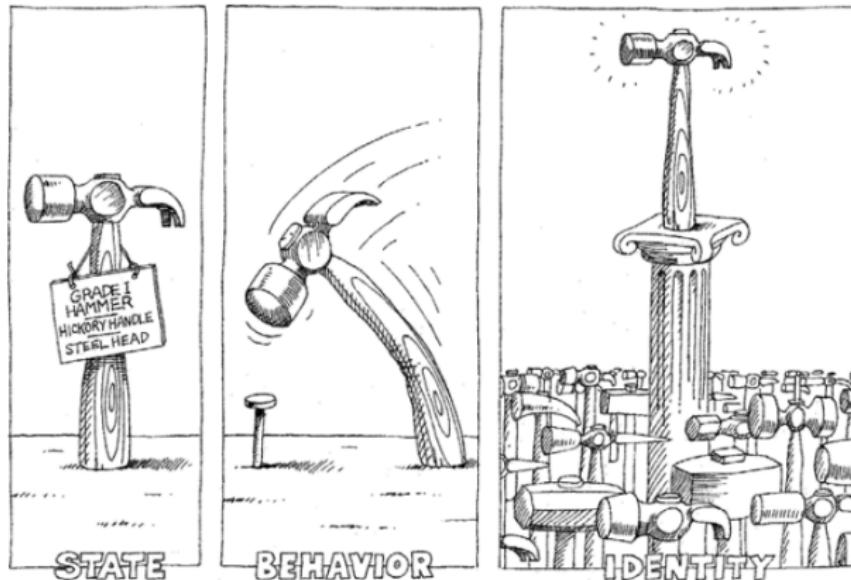
Validity
Constraints
Elevator
Space

BEHAVIOR

Common
Operations
Roles and
Responsibilities

IDENTITY

Display Object
Summary



State, Behavior and Identity of a Hammer Object

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Module Summary

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

- An object has State, Behavior and Identity
- State is defined by the values of properties
- Behavior is defined by the collection of operations
- Identity provides distinguishability
- An object occupies space, either in physical world, or in computer memory



Instructor and TAs

Module 11

Partha Pratim
Das

Objectives &
Outline

STATE
Validity
Constraints
Elevator
Space

BEHAVIOR
Common
Operations
Roles and
Responsibilities

IDENTITY
Display Object
Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

Module 12: Object Oriented Analysis & Design

Relationships among Objects

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

- Understanding Relationships among Objects



Module Outline

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization

Aggregation
Examples

Temperature
Controller

LMS

Summary

- Relationship / Associations among objects
- Links
 - Message Flow
 - Roles
 - Visibility
 - Synchronization
- Aggregation
- Examples
 - Temperature Controller
 - LMS



Relationships among Objects

Module 12

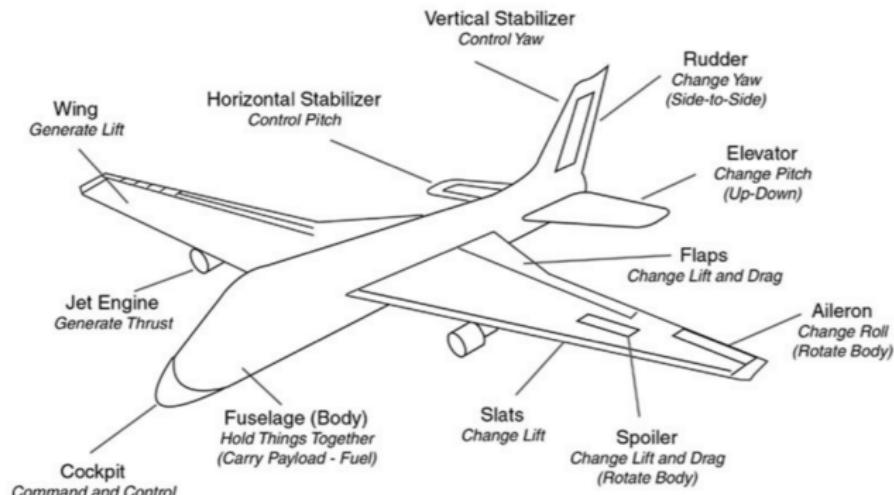
Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS
Summary

Objects contribute to the behavior of a system by collaborating with one another



All the components of an airplane have an inherent tendency to fall to earth, only their collaborative efforts enable it to fly and stave off any outcome of falling



Relationships among Objects

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization

Aggregation
Examples
Temperature
Controller
LMS

Summary

The two important object relationships in context of object-oriented analysis and design are:

- Links
- Aggregation



Links

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

A link is a physical or conceptual connection between objects

- An object collaborates with other objects through its links to these objects
- A Link denotes the specific association with the help of *passing messages* through which
 - One object uses/invokes/applies the services of another object
 - One object navigates to another object



Notation

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples

Temperature
Controller
LMS

Summary

- A link is depicted by a line between two objects
- The direction of the line, depicts which objects, invokes the services of the other object
 - If there is no direction, then the link is bidirectional, both the objects invokes each other's services
- A message can be unidirectional or bidirectional in a link
- A message is represented with a small directed line and label to define the message



Object A invokes the services of Object B



A Flow Control System

Module 12

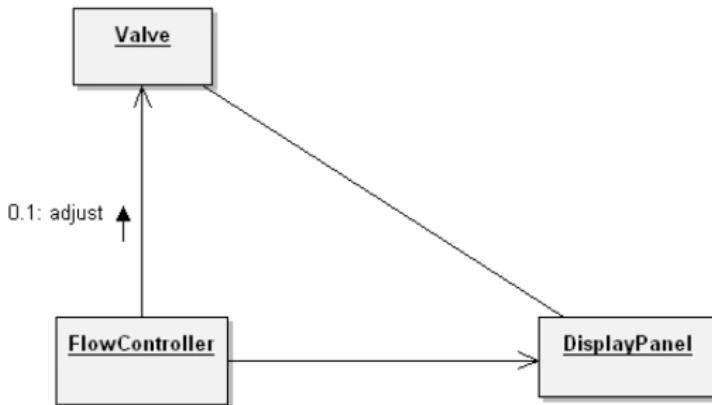
Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary



Links in a Flow Control System

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)

- The flow control system has three objects primarily: **FlowController**, **Valve** and **DisplayPanel**
- The **FlowController** object has a link to a **Valve** object
- The **FlowController** object has a link to the **DisplayPanel** object
- The **Valve** object has a link to the **DisplayPanel** object



Understanding Messages between Objects

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS
Summary

Message Exchange Between **FlowController** and **Valve**:

- **FlowController** invokes operations on the **Valve** object, but the flow of message can be bidirectional like request and response messages
- When **FlowController** invokes the operation *adjust* (message *Adjust sent*) on the **Valve** object to update the settings, then data flows from **FlowController** to **Valve**
- When **FlowController** invokes the operation, *isClosed* on the **Valve** object, to check if the **Valve** is closed, then the **Valve** object sends the response, hence data flows from **Valve** to **FlowController**



Roles of Participating Objects in a Link

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization

Aggregation
Examples

Temperature
Controller

LMS

Summary

- The **FlowController** object operates on the **Valve** object and **DisplayPanel** Object (to change settings or display string respectively); whereas the **Valve** and **DisplayPanel** objects do not operate on the **FlowController** object. Hence it is known as the **Controller** or **Active Object**
- **Separation of concerns** is achieved by the **FlowController** object
- The **DisplayPanel** object, does not operate on other objects; it is only operated on by other objects. Hence it is known as the **Server**
- The **Valve** object, can operate on **DisplayPanel** and get operated by **FlowController**. Hence it is known as the **Proxy**



Visibility

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility

Synchronization
Aggregation

Examples
Temperature
Controller
LMS

Summary

If the **FlowController** invokes operations on the **Valve** object, the **Valve** object should be visible to the **FlowController** object

We need to check the visibility across links because the scope and access of the objects on each side of a link are dependent

Types of visibility are:

- public
- package
- protected
- private



Synchronization

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility

Synchronization

Aggregation
Examples

Temperature
Controller

LMS

Summary

Whenever one object passes a message to another across a link, the two objects are said to be synchronized – they are *Active* and *Passive* Objects.

Object Type Illustrative Behavior

Active

- An object which instigates an interaction in a thread
- It can change its state by itself and send a message to other object if necessary
- It is responsible for handling control to other objects
- It is referred to as a **Client**

Passive

- An objects which passively waits for the message to be processed
- It is activated when it receives a message from other object
- It waits for another object that requires its services
- It is referred to as a **Server**



Synchronization

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

For objects with multiple threads of control, synchronization needs to ensure mutual exclusion to resolve issues of concurrency

Methods of concurrency used for a link between an active and a passive object:

Sequential

- Passive object linked to a single active object at a time
- Active objects must coordinate outside the object so that only one flow is in the object at a time (*function invocation*)
- For multiple flows of control, the integrity is not guaranteed
- **One Printer (passive) – one Print Task (active)**

Guarded

- The invoking object (Client) coordinates for mutual exclusion
- Integrity is guaranteed for multiple flows of control – serialize all calls to all of the object's guarded operations
- Exactly one operation at a time can execute on the object, reducing this to sequential semantics (**Beware of deadlock!**)
- **One Printer – one Print Task uses a queue to serialize**

Concurrent

- The invoked object (Server) coordinates for mutual exclusion
- Integrity is guaranteed for multiple flows of control – multiple flows access disjoint sets of data or only read data (**Crafty Design**)
- **One Printer uses a Spooler – multiple Print Tasks**



Aggregation

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

- **Links denote peer-to-peer or client/server relationships**
- **In contrast, Aggregation denotes a whole/part hierarchy, with the ability to navigate from the whole (or aggregate) to its parts, that is, contained objects**
- Aggregation may or may not denote physical containment
 - **HAS_A (Strong Aggregation):** Physical containment is necessary – Airplane & its Engine, Car & its Wheels, Book & its Chapters, Chapter & its Sections, etc.
 - **HAS (Weak Aggregation):** Conceptual aggregation, Physical containment is not present – Shareholders & Shares, Library & Users, etc.

An Aggregation is a specialized form of Association (Link)

- **Aggregation** is sometimes better because it encapsulates parts as secrets of the whole
- **Links** are sometimes better because they permit looser coupling among objects



A Temperature Controller System

Module 12

Partha Pratim
Das

Objectives &
Outline

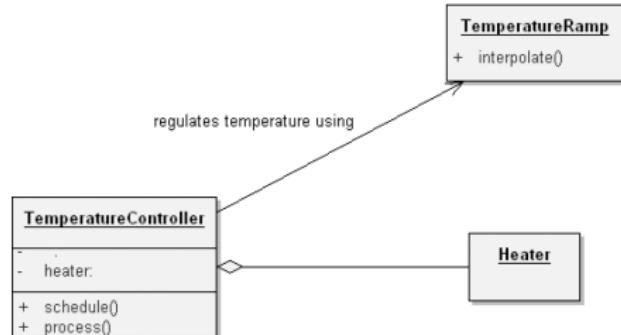
Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization

Aggregation
Examples

Temperature
Controller
LMS

Summary



- The temperature control system has two primary objects **TemperatureController** and **TemperatureRamp**
- The link between the two objects denote message flow
- The **Heater** object is a component of **TemperatureController**, hence it is a sub part and the **TemperatureController** is the whole
- The whole component **TemperatureController** will have a link to its aggregate **Heater**, to send or receive messages



Aggregation in LMS

Module 12

Partha Pratim
Das

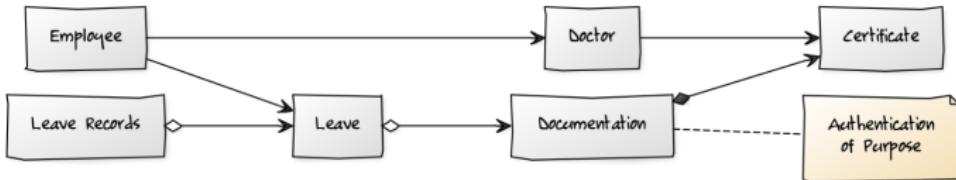
Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples

Temperature
Controller
LMS

Summary



Drawn using: <http://yuml.me/diagram/scruffy/class/draw>

- **Employee** Controller Object
 Sends `applyLeave()` message to `Leave`
 Sends `requestCert()` message to `Doctor`
- **Doctor** Proxy Object
 Receives `requestCert()` message from `Employee`
 Sends `issue()` message to `Certificate`
- **Certificate** Server Object
 Receives `issue()` message from `Doctor`
- **Leave Records** Weak Aggregation of `Leaves`
- **Leave** Weak Aggregation of `Documentation`
- **Documentation** Strong Aggregation of `Certificates`



Module Summary

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

- Object collaborate with each other through message passing. This communication channel is known as a link
- Using the link, the objects can invoke services on the other objects, and navigate to contained objects



Instructor and TAs

Module 12

Partha Pratim
Das

Objectives &
Outline

Relationships
/ Associations

Links
Message Flow
Roles
Visibility
Synchronization
Aggregation
Examples
Temperature
Controller
LMS

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

Module 13: Object Oriented Analysis & Design

Nature of a Class: Interface and Implementation

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

- Understanding Interface and Implementation of a Class



Module Outline

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

- Design by Contract
- Interface
 - Stack
- Visibility
- Implementation
 - Stack



What is a Class?

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

Whereas an object is a concrete entity that exists in time and space, a class represents only an abstraction, the "essence" of an object, as it were

- For a class **Faculty**, objects may be:
 - {Partha Pratim Das, Professor, CSE}
 - {Prabir Kumar Biswas, Professor, ECE}
 - {Shyamal Das Mondal, Assistant Professor, CET}
- Class **Faculty** abstracts – *Name, Designation, and Department*

A class is a set of objects share a common structure, common behavior, and common semantics

A single object is simply an instance of a class



What is a Class?

Module 13

Partha Pratim
Das

Objectives &
Outline

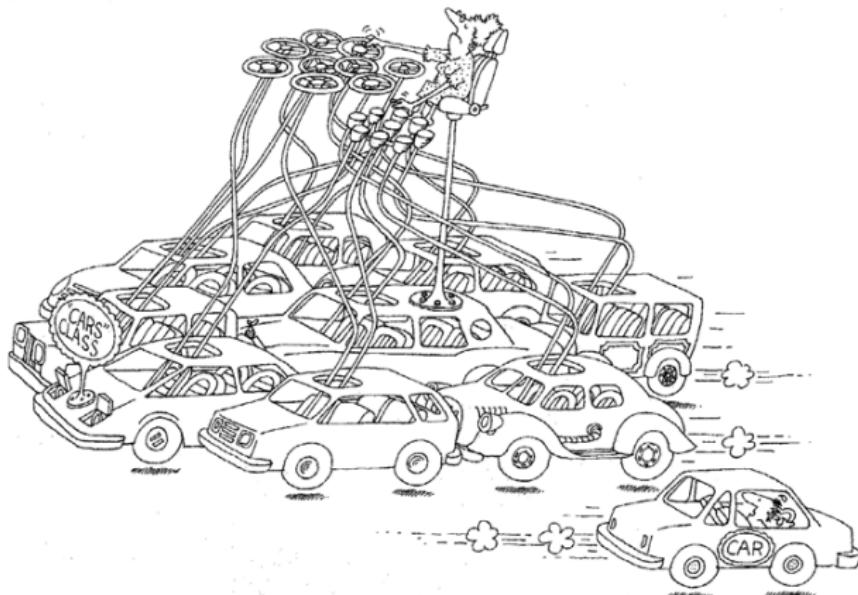
Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary



A class represents a set of objects with a common structure & a common behavior

Source: Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007)



What is a not Class?

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

- An object is not a class
- Objects that share no common structure and behavior cannot be grouped in a class because, by definition, they are unrelated except by their general nature as objects
 - Today's newspaper, Bajrangi Bhaijaan movie, State of West Bengal – are all objects that cannot be put together in a class
- While anything can be a class, the attributes of a complex system should be carefully crafted in designing a class:
 - Hierarchic Structure
 - Relative Primitives
 - Separation of Concerns
 - Common Patterns
 - Stable Intermediate Forms



Design by Contract (DbC)

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

- Design by Contract¹ is a metaphoric approach to design based on how elements of a software system collaborate with each other on the basis of mutual obligations and benefits
- The metaphor comes from business life, where a client and a supplier agree on a contract that defines for example that:
 - The supplier must provide a certain product (obligation) and is entitled to expect that the client has paid its fee (benefit)
 - The client must pay the fee (obligation) and is entitled to get the product (benefit)
 - Both parties must satisfy certain obligations, such as laws and regulations, applying to all contracts
- It has applications in:
 - Software design process
 - Inheritance with specific reference to dynamic binding
 - Exception handling
 - Automatic Software documentation

Source: Design by contract: https://en.wikipedia.org/wiki/Design_by_contract

¹The term was coined by Bertrand Meyer in connection with his design of the Eiffel programming language and first described in various articles starting in 1986



Design by Contract

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

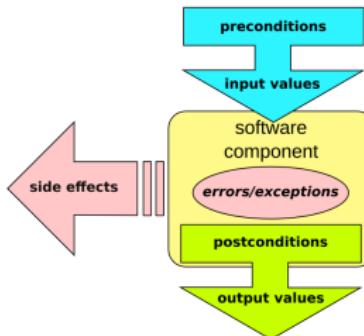
Interface
Stack

Visibility

Implementation
Stack

Summary

- For a class in OOP with methods:
- **Precondition** to be guaranteed on entry by any client that invokes the message – an obligation for the client, and a benefit for the server (the method itself), as it frees it from having to handle cases outside of the precondition
 - **Postcondition** is guaranteed on exit from the routine – an obligation for the server, and obviously a benefit (invoking the message) for the client
 - Maintain a certain property, assumed on entry and guaranteed on exit: the class **invariant**



A Design by Contract scheme

Source: *Design by contract*: https://en.wikipedia.org/wiki/Design_by_contract



Interface

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

The Interface of a class

- Provides its **Outside View**
- Emphasizes the abstraction while hiding its structure and the secrets of its behavior

The interface applies to all instances of a class and comprises:

- **Methods – publicly visible**
- **Contracts for the Methods**



Interface: Example of Stack

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

Consider as Stack class as a LIFO container. The interface may be:

Method	Precondition	Postcondition	Side-effect
Push	Valid object	Object on top	May acquire more memory
Pop	Stack not empty	Top object removed	May release some memory
Top	Stack not empty	Top object returned	
		State of stack unchanged	
Empty	Nil	State of stack unchanged	

An alternate interface may be:

Method	Precondition	Postcondition	Side-effect
Push	Valid object	Object on top	May acquire more memory
Pop	Nil	Top object removed	Throws exception, if stack is empty
Top	Nil	Top object returned	May release some memory
		State of stack unchanged	Throws exception, if stack is empty
Empty	Nil	State of stack unchanged	

Yet another interface may be:

Method	Precondition	Postcondition	Side-effect
Push	Valid object	Object on top	May acquire more memory
Pop	Nil	Top object removed	Throws exception, if low on memory
Top	Nil	Top object returned	Throws exception, if stack is empty
		State of stack unchanged	May release some memory
Empty	Nil	State of stack unchanged	Throws exception, if stack is empty



Visibility

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

We can divide the interface of a class into four parts:

- ① **Public**: a declaration that is accessible to all clients
- ② **Protected**: a declaration that is accessible only to the class itself and its subclasses
- ③ **Private**: a declaration that is accessible only to the class itself
- ④ **Package**: a declaration that is accessible only by classes in the same package

Visibility enforces grades of Encapsulation in a class



Visibility in Different OOP Languages

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

Language	Visibility			
	public	private	protected	others
C++	data members & member functions	data members & member functions	data members & member functions	friend: function & class
Java	fields & methods	fields & methods	fields & method	package: restriction by package
C#	variables & methods	variables & methods	variables & methods	internal: restriction by project
Python	member variables & methods	member variables	member variables	NA
Ada	declarations	declarations	NA	NA
Smalltalk	methods	instance variables	NA	NA
Object Pascal	fields & operations	NA	NA	NA



Implementation

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack
Summary

The **Implementation** of a class

- Deal with its **Inside View** which encompasses the secrets of its behavior.
- Consists primarily of the implementation of all of the operations defined in the interface of the class

In an implementation:

- The state of an object must have some representation – *typically expressed as constant and variable declarations placed in the protected or private part of a class's interface*
- The representation common to all instances of a class is encapsulated – changes to this representation do not functionally affect any outside clients



Implementation: Example of Stack

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

The **Stack** class may be implemented, say in C++, as:

- A static array and an index marker (top)
- A dynamic array and an index marker (top)
- A linked list and a header as top marker
- An STL vector and an index marker (top)

Any of the three interfaces discussed earlier may be implemented in any of the above four (or other) ways



Module Summary

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

- Introduced the notions of Design by Contract as a powerful design paradigm
- Discussed the Interface and Implementation of a Class and the use of Visibility for Encapsulation



Instructor and TAs

Module 13

Partha Pratim
Das

Objectives &
Outline

Design by
Contract

Interface
Stack

Visibility

Implementation
Stack

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

Module 14: Object Oriented Analysis & Design

Relationships among Classes

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- Understanding the Relationships among Classes



Module Outline

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- Relationships among Classes
- Association
- Inheritance
 - Polymorphism
- Aggregation
- Dependencies



Relationships of Classes

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary



Daisy



Red Rose



Yellow Rose



Petals



Ladybugs



Aphids

- A daisy is a kind of flower
- A rose is a (different) kind of flower
- Red roses and yellow roses are both kinds of roses
- A petal is a part of both kinds of flowers
- Ladybugs eat certain pests such as aphids, which may be infesting certain kinds of flowers



Relationships of Classes

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

Class

- A daisy is a kind of flower
- A rose is a (different) kind of flower
- Red roses and yellow roses are both kinds of roses
- A petal is a part of both kinds of flowers
- Ladybugs eat certain pests such as aphids, which may be infesting certain kinds of flowers

Relationship

Sharing connection – daisies and roses are both kinds of flowers – bright colored petals, fragrance, etc.

Daisy IS_A Flower

Sharing connection – daisies and roses are both kinds of flowers ...

Rose IS_A Flower

Semantic connection – red roses and yellow roses are more alike than are daisies & roses

Red Rose IS_A Rose, Yellow Rose IS_A Rose

Semantic connection – daisies and roses are more closely related than are petals & flowers

Flower HAS_A Petal

Symbiotic connection – Ladybugs protect flowers from certain pests

Semantic Dependency

Are Roses and Candles related? – Both decorate dinner tables

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Association

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies
Summary

• Semantic Dependencies

- Most general and most semantically weak
- Bidirectional by default
- Often refined over the analysis process



Early relationship



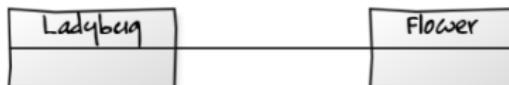
Refined to IS_A



Early relationship



Refined to HAS_A



Early relationship

Refined to ?



Association

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

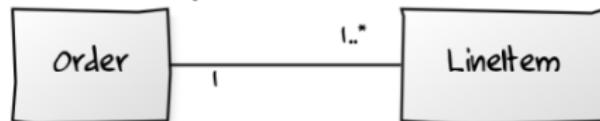
Summary

- Multiplicity of Relationships (Cardinality)

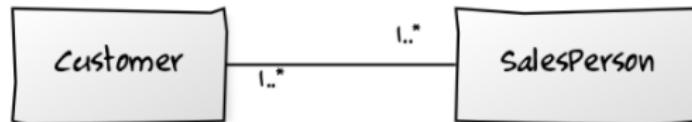
- One-to-one



- One-to-many



- Many-to-many





Inheritance (IS_A)

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polyorphism

Aggregation
(HAS_A)

Dependencies

Summary

- Generalization / Specialization relationships

- Say, we model **Daisy IS_A Flower**
- Daisy** will **inherit** the properties of **Flower**, and have some more of its own
- Flower** is the **Generalization**
- Daisy** is the **Specialization**
- Depicted as:



- Semantically most interesting
- Can *delegate* behavior to related objects
- Comes in a number of flavors
 - Single / Multilevel / Hierarchical Inheritance
 - Multiple Inheritance
 - Hybrid Inheritance



Single / Hierarchical Inheritance

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

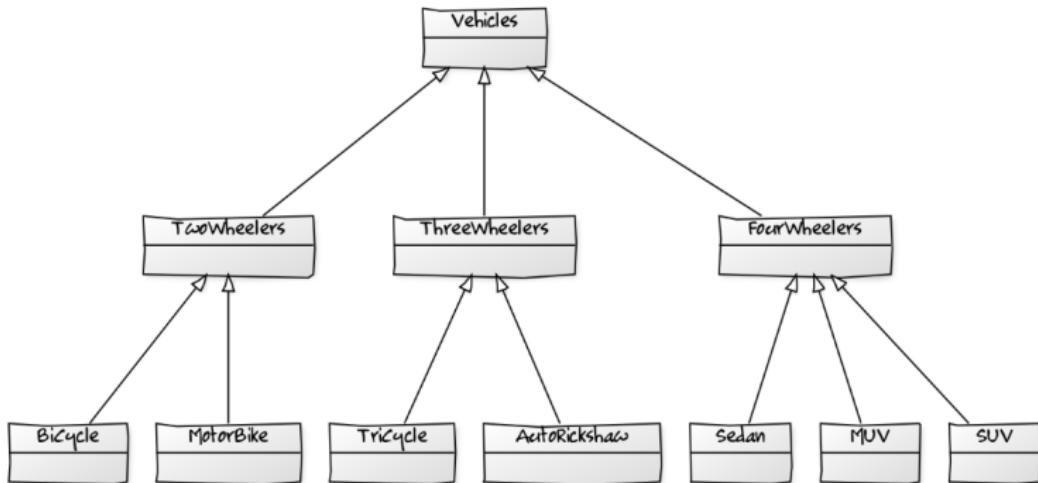
Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary



- **TwoWheelers** is a **superclass** of **BiCycle** and **subclass** of **Vehicles**
- **BiCycle**, **MotorBike**, **TriCycle**, **AutoRickshaw**, **Sedan**, **SUV**, and, **MUV** are **Leaf** or **Concrete** classes
- **TwoWheelers**, **ThreeWheelers**, **FourWheelers**, and **Vehicles** are **Abstract** classes – **they can have no instance**



Single / Hierarchical Inheritance

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- A subclass that augments its superclasses is said to use **inheritance for extension**
 - Eagle IS_A Bird
 - Sedan IS_A FourWheeler
- A subclass that constrains the behavior of its superclasses is said to use **inheritance for restriction**
 - Ostrich IS_A Bird
 - Square IS_A Rectangle



Polymorphism

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

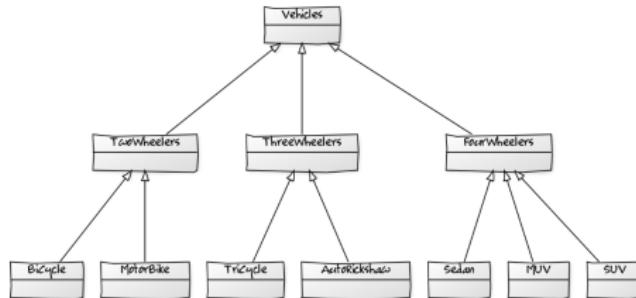
Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary



- Suppose, all classes above have a method `getNumberOfWheels()`
 - For `TwoWHEELERS` (and its subclasses) it will return 2
 - For `ThreeWHEELERS` (and its subclasses) it will return 3
 - For `FourWHEELERS` (and its subclasses) it will return 4
 - This is called **Polymorphism** – one name, but multiple behavior
 - **What will it return for Vehicles?**
- For another method `drive()`, the classes will again have different behavior – need different skills to ride a bike / auto / car; need different licenses, etc.
 - **Can we implement `drive()` for TwoWHEELERS**



Multiple Inheritance

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

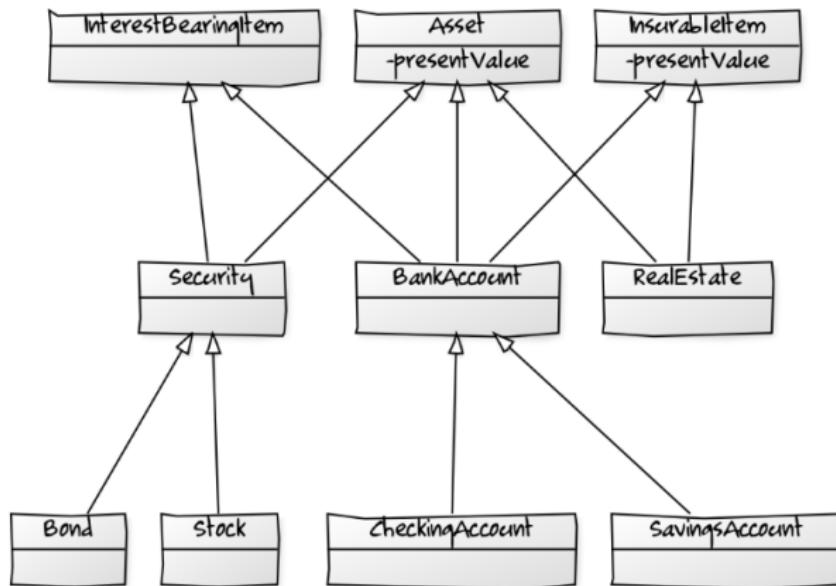
Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary



- More than one **superclass** for a **subclass**
- RealEstate IS_A Asset, InsurableItem**



Multiple Inheritance: Issues

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- How do we deal with **name collisions** from different superclasses?
 - `presentValue` in both `InsurableItem` and `Asset` – what happens to `RealEstate`?
 - Clashes may introduce ambiguity in the behavior of the multiply inherited subclass
 - Resolution by Language Semantics:
 - ① Regard such a clash as illegal and reject the compilation of the class
 - ② Regard the same name introduced by different classes as referring to the same attribute
 - ③ Permit the clash but require that all references to the name fully qualify the source of its declaration (C++ style)



Multiple Inheritance: Issues

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

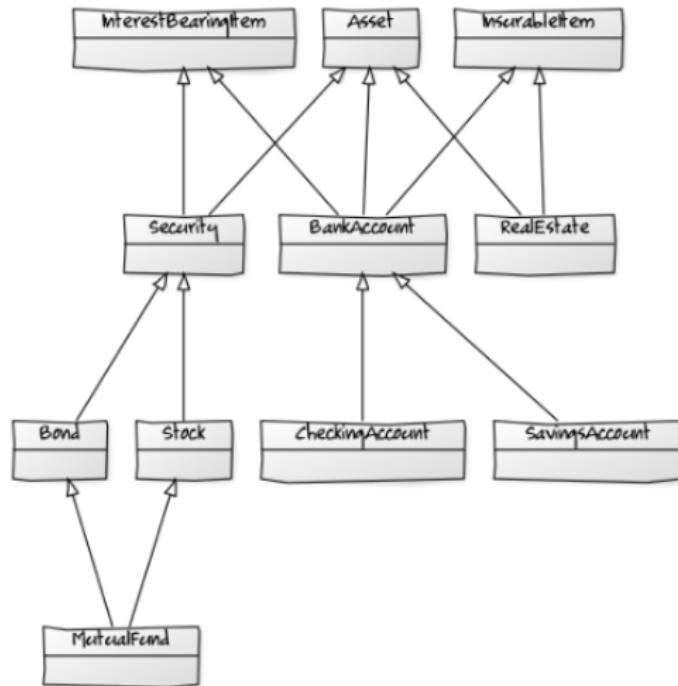
Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- How do we handle **repeated inheritance?** [Diamond]





Multiple Inheritance: Issues

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- How do we handle **repeated inheritance?**

- Resolution by Language Semantics:
 - ① Treat occurrences of repeated inheritance as illegal
 - ② Permit duplication of superclasses but require the use of fully qualified names to refer to members of a specific copy
 - ③ Treat multiple references to the same class as denoting the same class
- Different languages handle this approach differently



Aggregation (HAS_A)

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- Whole / Part relationships

- Say, we model **Flower HAS_A Petal**
- Flower contains many Petals**
- Flower is the Whole, Petal is the Part**
- Depicted as:



- Physical Containment – **Composition / Strong Aggregation**

- Member relationship

- Say, we model **Library HAS Users**
- Library enrolls many Users**
- Library does not contain the Users**
- Depicted as:



- Conceptual Containment – **Weak Aggregation**



Dependencies

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- All other kinds of dependencies
- A dependency indicates that an element on one end of the relationship, in some manner, depends on the element on the other end of the relationship
- Example:





Module Summary

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

- We have discussed various relationships among Classes. Specifically,
 - Association
 - Inheritance
 - Aggregation
 - Dependencies
- Most of these relationships are closely similar to the relationships among Objects



Instructor and TAs

Module 14

Partha Pratim
Das

Objectives &
Outline

Association

Inheritance
(IS_A)

Polymorphism

Aggregation
(HAS_A)

Dependencies

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter
Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

Module 15: Object Oriented Analysis & Design

How to Build Quality Classes and Objects

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

- Understanding how to Build Quality Classes and Objects



Module Outline

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

- Measuring the Quality of an Abstraction
- Choosing Operations
 - Functional Semantics
 - Time and Space Semantics
- Choosing Relationships
 - The Law of Demeter
 - Mechanisms and Visibility
- Choosing Implementations
 - Representation
 - Packaging



Measuring the Quality of an Abstraction

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations
Functional
Semantics
Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter
Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging
Summary

How can one know if a given class or object is well designed? Use measures:

① Coupling

- Measure of the strength of association established by a connection from one module to another
- Low inter-class coupling desirable
- Inheritance is highly coupled!

② Cohesion

- High intra-class cohesion desirable

③ Sufficiency

- The class or module captures enough characteristics of the abstraction to permit meaningful and efficient interaction
- Leads to minimalist design

④ Completeness

- The interface of the class or module captures all of the meaningful characteristics of the abstraction
- No aspects of the abstraction is missed out

⑤ Primitiveness

- Primitive operations are those that can be efficiently implemented



Choosing Operations

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

Crafting an Interface involves the decisions about:

- **Functional Semantics**
- **Time and Space Semantics**



Choosing Operations: Functional Semantics

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations
Functional
Semantics
Time and Space
Semantics

Choosing
Relationships
The Law of
Demeter
Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

- Too fine – Too coarse trade-off

A good designer knows how to find the appropriate balance between too much contracting, which produces fragmentation, and too little, which yields unmanageably large modules

- How to place a method?

- Reusability:** Would this behavior be more useful in more than one context?
- Complexity:** How difficult is it to implement the behavior?
- Applicability:** How relevant is the behavior to the type in which it might be placed?
- Implementation Knowledge:** Does the behavior's implementation depend on the internal details of a type?



Choosing Operations: Time and Space Semantics

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

- Specify the amount of time an operation should take
- Specify the amount of space an operation should take
- This semantics is typically analyzed for – best, average, and worst cases
- Consider the need of Synchronization for message passing (objects may be in multiple threads – not simple function invocation)



Choosing Relationships

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter
Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

- Choosing the relationships among classes and among objects is linked to the selection of operations
- For example, *if we decide that object X sends message M to object Y, then either directly or indirectly, Y must be accessible to X; otherwise, we could not name the operation M in the implementation of X*
- The following principles help in the decisions:
 - **The Law of Demeter**
 - **Mechanisms and Visibility**



Choosing Relationships: The Law of Demeter

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter
Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

- The Law of Demeter:

The methods of a class should not depend in any way on the structure of any class, except the immediate (top-level) structure of their own class. Further, each method should send messages to objects belonging to a very limited set of classes only

- How to analyze the Class Inheritance Hierarchy Structure?

- **Wide and Shallow:** Usually represent forests of free-standing classes that can be mixed and matched – Forests of classes are more loosely coupled, but they may not exploit all the commonality that exists
- **Narrow and Deep:** Usually represent trees of classes that are related by a common ancestor – Long chain of inheritance is tightly coupled
- **Balanced:** Between the above two extremes
- Similar trade-offs among inheritance, aggregation, and dependency relationships to be analyzed
- Should class **Car inherit, contain, or use** classes – **Engine & Wheel?**

Inheritance is appropriate if every instance of B may also be viewed as an instance of A. The client relationship is appropriate when every instance of B simply possesses one or more attributes of A



Choosing Relationships: Mechanisms and Visibility

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

- Deciding on the relationship among objects is a matter of designing the mechanisms whereby these objects interact
- A **Passenger** intends to **board** a **Bus**. Where should be **board** operation go?
 - **Passenger** – **Passenger** must be visible to **Bus**, because **Passenger** must know which **Bus** she is getting into
 - **Bus** – **Bus** must be visible to **Passenger**
 - Both – there must be mutual visibility
 - Further, visibility relationship between **Bus** and **Driver** must be clear while the same between **Passenger** and **Driver** need not be there



Choosing Implementations

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

- Next consider the inside view
- This perspective involves two different decisions:
 - **Representation for a class or object**
 - **Placement of the class or object in a module**



Choosing Implementations: Representation

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter
Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

- Representation should be one of the encapsulated secrets of the abstraction
- Possible to change the representation (like, to alter the time and space semantics) without violating any of the functional assumptions that clients may have made

The choice of representation is often a fairly difficult one, and it is not uniquely determined by the facilities available. It must always be taken in light of the operations that are to be performed upon the data

- Time or Space? What has priority?
- Search or Insert / Delete? What has priority?
- Compute or Cache? What has priority?
- ...



Choosing Implementations: Packaging

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

- How to put Classes and Objects in Modules?
- Visibility and Information hiding trade off

Applying this principle is not always easy. It attempts to minimize the expected cost of software over its period of use and requires that the designer estimate the likelihood of changes. Such estimates are based on past experience and usually require knowledge of the application area as well as an understanding of hardware and software technology

- Many nontechnical factors – matters of reuse, security, and documentation, ...



Module Summary

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations

Representation
Packaging

Summary

- We have defined the measures of quality of an abstraction
- We introduced guidelines for Choosing Operations, Relationships, and Implementations



Instructor and TAs

Module 15

Partha Pratim
Das

Objectives &
Outline

Measures of
Quality

Choosing
Operations

Functional
Semantics

Time and Space
Semantics

Choosing
Relationships

The Law of
Demeter

Mechanisms and
Visibility

Choosing Im-
plementations
Representation
Packaging

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes
Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Module 16: Object-Oriented Analysis & Design

How to Identify Classes & Objects?

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory

Identification
Key Abstraction

How to
Apply?

Summary

- Understand the importance and difficulties of classification
- Identify the approaches of classification and identification
- Prepare to apply the theory in practice



Module Outline

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory

Identification
Key Abstraction

How to
Apply?

Summary

- What is Classification
 - Importance of proper classification
 - Difficulties of classification
- Identification of Classes
 - Classification
 - Structural Clustering
 - Conceptual Clustering
 - Prototyping
 - Identification
 - Key Abstractions
- How to Apply?



Classification

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Classification is the means whereby we order knowledge





Importance of Proper Classification

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

- Classification helps to identify generalization, specialization, and aggregation hierarchies among classes
- Classification guides in making decisions about modularization
- Coupling and cohesion also indicate a type of sameness
- Classification also plays a role in allocating processes to processors



Difficulty of Classification

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

- There are potentially at least as many ways of dividing up the world into object systems as there are scientists to undertake the task
- Intelligent classification requires a tremendous amount of creative insight
 - Why is a laser beam like a goldfish? . . . because neither one can whistle



Difficulty of Classification - Example

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

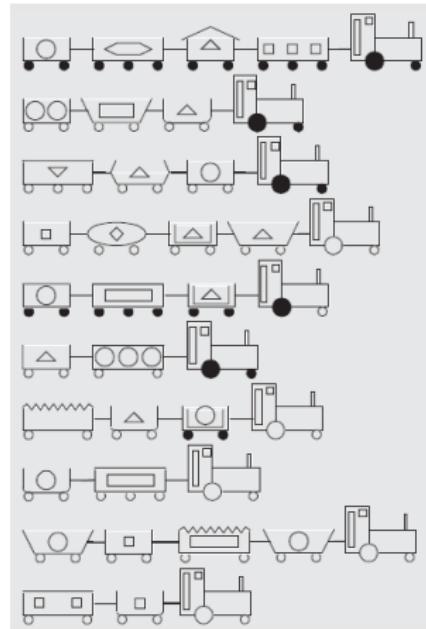
Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Create meaningful groups by arranging trains given below





Difficulty of Classification - Example

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

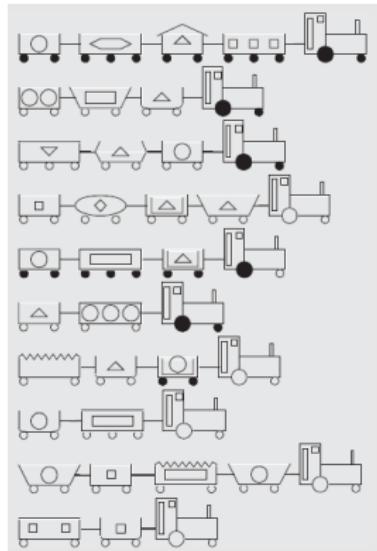
Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Divide objects into disjoint sets depending on the presence or absence of a particular property

- You might create three groups: black wheels, white wheels, and black and white wheels.
- Classification by the length of the train, forming three groups: trains with two, three, and four bogies
- 93 different classifications are possible
- As in real life, there is no *right* answer





Approaches for Identification of Classes

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

- Clustering
 - Structural Clustering
 - Conceptual Clustering
 - Prototype Theory
- Identification
 - Key Abstractions



Structural Clustering

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

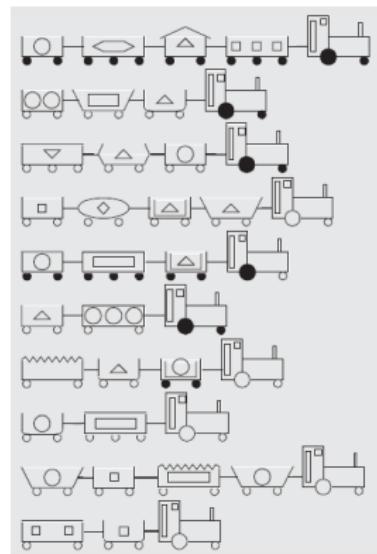
Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Divide objects into disjoint sets depending on the presence or absence of a particular property

- Three groups: black wheels, white wheels, and black and white wheels
- Three groups: trains with two, three, and four bogies





Conceptual Clustering

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

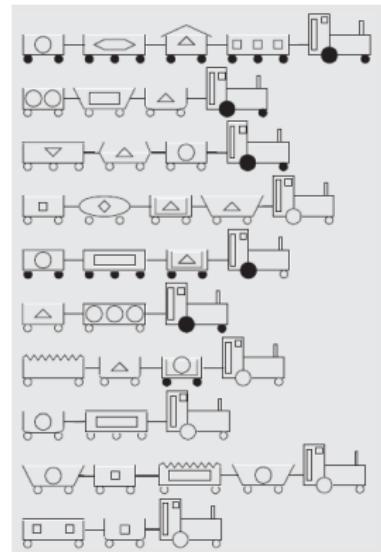
Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction
How to
Apply?

Summary

Attempts to explain how knowledge is represented

- Let us change the requirements as in real life
- Suppose
 - circles represent toxic chemicals
 - rectangles represent lumber, and
 - all other shapes of loads represent passengers
- Try classifying the trains again





Conceptual Clustering

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

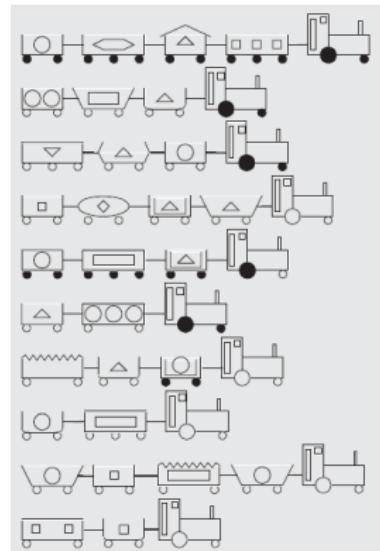
Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

New knowledge changes the classification

- Classify trains according to whether or not they carried toxic loads
- Classify trains: Passenger train and Goods train (Subclass - Toxic and Non Toxic)
- More knowledge about a domain makes it easier to achieve an intelligent classification





Prototype Theory

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Some abstractions neither have clearly bounded properties nor concepts. Example:

- Game
 - No common properties shared by all games
 - The category of games is united by what Wittgenstein calls family resemblances
 - New kinds of games introduced, provided that they resembled previous games in appropriate ways

This approach is called *Prototype Theory*

- A class of objects is represented by a prototypical object, and
- An object is considered to be a member of this class if and only if it resembles this prototype



Identification of Key Abstraction

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

- Key abstraction are classes or objects that forms part of the vocabulary of the problem domain
- Give boundaries to the problem
- Highlight the things that are in the system
- Identification of key abstractions is highly domain-specific
- Identification involves two processes:
 - **Discovery** - A customer using an automated teller speaks in terms of accounts, deposits, and withdrawals; these words are part of the vocabulary of the problem domain
 - **Invention** - A developer of such a system uses these same abstractions but must also introduce new ones, such as databases, screen managers, lists, queues, and so on



How to Apply?

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

- ① Identify classes and objects first according to the properties relevant to the particular domain. Here, the focus is on identifying
 - Structures and behavior
- ② If this approach fails to yield a satisfactory class structure, next consider clustering objects by concepts (or refining our initial domain-based classification by concepts). Here, the focus is on
 - Behavior of collaborating objects
- ③ If either of these two approaches fails, consider classification by association, through which clusters of objects are defined according to how closely each resembles some prototypical object
- ④ Once classified, try to identify Key Abstractions



Module Summary

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification

Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory

Identification
Key Abstraction

How to
Apply?

Summary

- We understand that the identification of classes and objects is a challenging part of OOAD as there is no "perfect" classification, some classifications are better than others
- We learn approaches of classification and identification of key abstractions
- Understand how to apply them



Instructor and TAs

Module 16

Partha Pratim
Das

Objectives &
Outline

Classification
Importance
Difficulty

Identification
of Classes

Clustering
Structural
Clustering
Conceptual
Clustering
Prototype
Theory
Identification
Key Abstraction

How to
Apply?

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Module 17: Object-Oriented Analysis & Design

Identification of Classes, Objects & Relationships in LMS:

An Exploratory Exercise: Part 1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- Perform an exploratory exercise to identify potential classes, objects, and relationships for the Leave Management System (LMS)
- The following will be used as primary source of information for LMS (as received from the client)
 - Written English description of LMS (PDF document as shared)
 - Conversation between client and vendor (Tutorial video as uploaded)
- This will be a multi-module exercise spanning Module 17 to 20

Please keep these handy while you study these modules. We shall frequently refer to those



Module Outline

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- Input and Output Specification
- Informal English Description of LMS
- Identification of classes
 - Extraction of Nouns
 - Quality Check
 - Identification of Attributes
 - Structural Clustering
 - Quality Check
 - Behavioral Clustering
 - Quality Check
 - Identified Classes



Input and Output Specification

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

Input

Informal English Description ⇒

Output

- Classes
- Attributes
- Responsibilities

Class Name	
Attributes	Responsibilities
•	•
•	•
•	•



Informal English Description of LMS

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

A Company wants to manage the attendance and leave of its employees through LMS. The requirement specifications are:

- ① The company has three categories of employees:
 - *Executive*: Employees who work as individual contributors and report to a Lead.
 - *Lead*: Every Executive reports to a Lead who approves / regrets her / his leave. A Lead reports to the Manager.
 - *Manager*: Every Lead reports to the Manager who approves / regrets her / his leave. There is *only one* Manager.



Identification of Noun

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering

QC

Behavioral
Clustering
QC

Identified Classes

Summary

- Linguistic analysis of the problem description - extraction of **Nouns**
- Rules of thumb¹ :
 - Elimination of irrelevant terms
 - Elimination of names of values
 - Elimination of vague terms
 - Identification of attributes
 - Identification of operations
 - Elimination of terms which are in fact relationships

¹ Source: R.J. Abbott, Program Design by Informal English Descriptions, CACM, Vol.26, No.11, 1983



Identification of Noun

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering

QC

Behavioral
Clustering

QC

Identified Classes

Summary

A Company wants to manage the attendance and leave of its employees through LMS. The requirement specifications are:

- The company has three categories of employees:
 - Executive: Employees who work as individual contributors and report to a Lead.
 - Lead: Every Executive reports to a Lead who approves / regrets her / his leave. A Lead reports to the Manager.
 - Manager: Every Lead reports to the Manager who approves / regrets her / his leave. There is *only one Manager*.



Identification of Noun

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering

QC

Behavioral
Clustering

QC

Identified Classes

Summary

- The **company** has provisions for the following categories of **leave** associated with the respective **leave rules**:
 - *Casual Leave (CL)*:
 - 10 **CL's** are available in a **calendar year**. All **CL's** are credited to an **employee** on 01-Jan. For **employees** joining in the middle of the **year**, the number of **CL's** are prorated. **CL's** cannot be carried over to the next **calendar year**.
 - More than 2 **CL's** cannot be availed at a time. **CL's** cannot be clubbed with other **types of leave**. Total period of **absence** including **holidays** cannot be more than **4 days**. **Holidays** intervening the **absence** are not counted as **leave**.
 - **CL's** do not need **pre-approval**; but must be approved within 2 days of its availing.
 - ...



List of all Identified Nouns

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns

QC

Identification of
Attributes

Structural
Clustering

QC

Behavioral
Clustering

QC

Identified Classes

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			



Quality Check

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- The Metrics at this Stage

- ① **Coupling:** Very low
- ② **Cohesion:** Very low
- ③ **Sufficiency:** Unclear
- ④ **Completeness:** Unclear
- ⑤ **Primitiveness:** Unclear

- Actions required

- Identify Attributes
- Perform Structural Clustering



Attributes

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			

Noun used only as value should be treated as attribute

- Login ID
- Name
- Employee Code
- Personal details
- Designation
- Salary



Structural Clustering

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			

- People

- Manager, Lead, Executive
- SysAdmin

- Organizations

- Company

- Things (Documents)

- Medical Certificate, Parenthood Certificate



Structural Clustering

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			

- Events

- Holiday, Disciplinary Actions
- CL, ML, UL ...etc

- Temporal Nouns

- Date, Year, Time, Days, Calender year, Week, Month

- Incidental Nouns

- Leave Rule, Administrative Functions, Period



Structural Clustering

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- People
 - Manager, Lead, Executive
 - SysAdmin
- Organizations
 - Company
- Things (Documents)
 - Medical Certificate, Parenthood Certificate
- Events
 - Holiday, Disciplinary Actions, CL, ML, UL ...etc
- Temporal Nouns
 - Date, Year, Time, Days, Calender year, Week, Month
- Incidental Nouns
 - Leave Rule, Administrative Functions, Period



Quality Check

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- The Metrics at this Stage

- ① **Coupling:** **Low** (was Very low)
- ② **Cohesion:** **Low** (was Very low)
- ③ **Sufficiency:** Unclear (was Unclear)
- ④ **Completeness:** Unclear (was Unclear)
- ⑤ **Primitiveness:** Unclear (was Unclear)

- Actions required

- Perform Behavioral Clustering



Behavioral Clustering

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- What takes place in the system
 - Leaves – CL, ML, UL ...etc
- Initiator and Participant
 - Initiator – Lead and Manager; SysAdmin
 - Participant – Executive, Lead and Manager



Abstractions Summary from Linguistic Analysis

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- **Key Abstractions:** Leave, Employee, CL, EL, ...
- **Non-Key (Supporting) Abstractions:** Medical Certificate, Parenthood Certificate, Disciplinary Actions, ...
- **Incidental Abstractions:** Leave Status, Administrative Functions, ...
- **Attribute (Property) Abstractions:** name, personal details, employee code, ...



Quality Check

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- The Metrics at this Stage

- ① **Coupling:** Low (was Low)
- ② **Cohesion:** Moderate (was Low)
- ③ **Sufficiency:** Unclear (was Unclear)
- ④ **Completeness:** Very low (was Unclear)
- ⑤ **Primitiveness:** Very low (was Unclear)

- Actions required

- Decide on Classes (first level)



Resultant Classes

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- ① Company
- ② Employee
- ③ Executive
- ④ Lead
- ⑤ Manager
- ⑥ Leave
- ⑦ SL
- ⑧ ML
- ⑨ PL
- ⑩ CL
- ⑪ EL
- ⑫ DL
- ⑬ LWP
- ⑭ UL
- ⑮ SysAdmin



Module Summary

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

- We understand how can we identify classes and its attributes from an informal English description
- In the next module we will discuss the relation and hierarchy within the classes



Instructor and TAs

Module 17

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Extraction of
Nouns
QC

Identification of
Attributes

Structural
Clustering
QC

Behavioral
Clustering
QC

Identified Classes

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

Module 18: Object-Oriented Analysis & Design

Identification of Classes, Objects & Relationships in LMS: An Exploratory Exercise: Part 2

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- Perform an exploratory exercise to identify potential classes, objects, and relationships for the Leave Management System (LMS)
- The following will be used as primary source of information for LMS (as received from the client)
 - Written English description of LMS (PDF document as shared)
 - Conversation between client and vendor (Tutorial video as uploaded)
- This will be a multi-module exercise spanning Module 17 to 20

Please keep these handy while you study these modules. We shall frequently refer to those



Module Outline

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ties

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- Input and Output Specification
- Informal English description of LMS
- Identification of classes, their attributes and responsibilities from the description of LMS
- Resultant Classes of LMS with Responsibilities



Input and Output Specification

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

Input

Informal English Description ⇒

Output

- Classes
- Attributes
- Responsibilities

Class Name	
Attributes	Responsibilities
•	•
•	•
•	•



Informal English Description of LMS

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

A Company wants to manage the attendance and leave of its employees through LMS. The requirement specifications are:

- ① The company has three categories of employees:
 - *Executive*: Employees who work as individual contributors and report to a Lead.
 - *Lead*: Every Executive reports to a Lead who approves / regrets her / his leave. A Lead reports to the Manager.
 - *Manager*: Every Lead reports to the Manager who approves / regrets her / his leave. There is *only one* Manager.



Identification of Noun: RECAP (Module 17)

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

- Linguistic analysis of the problem description - extraction of **Nouns**
- Rules of thumb¹ :
 - Elimination of irrelevant terms
 - Elimination of names of values
 - Elimination of vague terms
 - Identification of attributes
 - Identification of operations
 - Elimination of terms which are in fact relationships

¹ Source: R.J. Abbott, Program Design by Informal English Descriptions, CACM, Vol.26, No.11, 1983



List of all Identified Nouns: RECAP (Module 17)

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			



Attributes: RECAP (Module 17)

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

- Login ID
- Name
- Employee Code
- Personal details
- Designation
- Salary
- ...



Quality Check: RECAP (Module 17)

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

- The Metrics at this Stage

- ① **Coupling:** Low (was Low)
- ② **Cohesion:** Moderate (was Low)
- ③ **Sufficiency:** Unclear (was Unclear)
- ④ **Completeness:** Very low (was Unclear)
- ⑤ **Primitiveness:** Very low (was Unclear)

- Actions required

- Decide on Classes (first level)



Resultant Classes: RECAP (Module 17)

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

- | | |
|-------------|------------|
| ① Company | ⑧ ML |
| ② Employee | ⑨ PL |
| ③ Executive | ⑩ CL |
| ④ Lead | ⑪ EL |
| ⑤ Manager | ⑫ DL |
| ⑥ Leave | ⑬ LWP |
| ⑦ SL | ⑭ UL |
| | ⑮ SysAdmin |



Identification of Responsibilities

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- We now use the Linguistic analysis of verbs and repeatedly refine these to identify responsibilities



Identification Verb

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

- Linguistic analysis of the problem description - extraction of verbs
- Rules of thumb²:
 - Which operations can be executed by a certain object
 - Not only the current requirements should be considered, but also re-usability should be taken into account
 - Which events are expected
 - Which objects can react to these events
 - Which other events are raised in turn

² Source: R.J. Abbott, Program Design by Informal English Descriptions, CACM, Vol.26, No.11, 1983



Identification Verb

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs

Verbs with
Classes

Results

QC

Summary

A Company **wants** to **manage** the attendance and leave of its employees through LMS. The requirement specifications are:

- ① The company has three categories of employees:
 - *Executive*: Employees who **work** as individual contributors and **report** to a Lead.
 - *Lead*: Every Executive **reports** to a Lead who **approves** / **regrets** her / his leave. A Lead **reports** to the Manager.
 - *Manager*: Every Lead **reports** to the Manager who **approves** / **regrets** her / his leave. There is *only one* Manager.



Identification Verb

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- The company has provisions for the following categories of leave associated with the respective leave rules:
 - *Casual Leave (CL):*
 - 10 CL's are available in a calendar year. All CL's are **credited** to an employee on 01-Jan. For employees **joining** in the middle of the year, the number of CL's are **prorated**. CL's **cannot be carried over** to the next calendar year.
 - More than 2 CL's **cannot be availed** at a time. CL's **cannot be clubbed** with other types of leave. Total period of absence including holidays **cannot be** more than 4 days. Holidays intervening the absence are not **counted** as leave.
 - CL's do not need pre-approval; but must be **approved** within 2 days of its availing.
 - ...



List of all Identified Verbs

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

Wants	Manage	Work	Report
Approve	Regret	Credit	Join
Prorate	Cross	En-cash	Paid
Allow	Send	Need	Become
Enjoy	Avail	Proceeding	Employ
Consider	Deduct	Provide	Request
Cancel	Check	Export	Revoke
Debit	Adjust	Perform	Hire
Fire	Generate	Leave	Can be Availed
		En-cashment	
Can be	Can't be	Can't be	Can't be
Clubbed	Availed	Carried forward	Clubbed
Can't be	Accumulated	Proposed for	Join Back
Continued	Up		
Doesn't Draw	Can be Revoked	Leave Credited	

Many extracted verbs are in derived forms – so we extract the unique stems



List of all Identified Verbs (Stem Only)

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs

Verbs with
Classes

Results

QC

Summary

Wants	Manage	Work	Report
Approve	Regret	Credit	Join
Prorate	Cross	En-cash	Pay
Allow	Send	Need	Become
Enjoy	Avail	Proceed	Employ
Consider	Deduct	Provide	Request
Cancel	Check	Export	Revoke
Debit	Adjust	Perform	Hire
Fire	Generate	Club	Carry forward
Continue	Accumulate	Propose	Join Back
Draw			



Relation of the Verbs with Classes

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

Class	Related Verbs / Responsibilities
Employee	<ul style="list-style-type: none">• Work• Record• Request• Cancel• Avail• Check• Export• Report• Approves• Regret
Casual Leave	<ul style="list-style-type: none">• Credit• Prorate• Approve• Cannot be availed• cannot be clubbed



Relation of the Verbs with Classes

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC
Summary

Class	Responsibilities
Employee	<ul style="list-style-type: none">• Work• Record• Request• Cancel• Avail• Check• Export• Report• Approves• Regret
Casual Leave	<ul style="list-style-type: none">• Credit• Prorate• Approve• Cannot be availed• cannot be clubbed

Blue responsibilities are core – common for all employees



Abstraction Summary from Linguistic Analysis

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- **Key Actions:** Report, Request (leave), Approve (leave), Regret (leave), ...
- **Non-Key (Supporting) Actions:** Can be clubbed, ...
- **Auxiliary Actions:** Perform, ...



Employee Classes with its Attributes and Responsibilities

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ties

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

Employee	
Attributes	Responsibilities
<ul style="list-style-type: none">• Name• Personal details• Designation• Employee code• Login id	<ul style="list-style-type: none">• Record Daily Attendance• Request for Leave• Cancel an Approved Leave not yet availed• Avail Leave (if approved)• Check / Export own Leave Status for a period

- Similarly attributes and responsibilities can be for Executive, Lead and Manager classes
- Try and complete the rest



Leave Classes with its Attributes and Responsibilities

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results

QC

Summary

Leave	
Attributes	Responsibilities
<ul style="list-style-type: none">• Type of Leave• Start Date• Duration of Leave• Employee Id	<ul style="list-style-type: none">• Validity Check• Accounting• •

- Similarly attributes and responsibilities can be for CL, EL, ML ...etc.
- Try and complete the rest



Quality Check

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- The Metrics at this Stage

- ① **Coupling:** Low (was Low)
- ② **Cohesion:** Moderate (was Moderate)
- ③ **Sufficiency:** Very low (was Unclear)
- ④ **Completeness:** Low (was Very low)
- ⑤ **Primitiveness:** Very low (was Very low)

- Actions required

- Explore Association



Module Summary

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

- We analyzed how can we identify classes and its attributes and responsibilities from a informal English description
- In the next module we explore the relation and hierarchy within the classes



Instructor and TAs

Module 18

Partha Pratim
Das

Objectives &
Outline

I/O Specs

Description of
LMS

Identification
of Classes

Identification
of Responsibil-
ities

Extraction of
Verbs
Verbs with
Classes

Results
QC

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Module 19: Object-Oriented Analysis & Design

Identification of Classes, Objects & Relationships in LMS: An Exploratory Exercise: Part 3

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy
Employee
Hierarchy

Summary

- Perform an exploratory exercise to identify potential classes, objects, and relationships for the Leave Management System (LMS)
- The following will be used as primary source of information for LMS (as received from the client)
 - Written English description of LMS (PDF document as shared)
 - Conversation between client and vendor (Tutorial video as uploaded)
- This will be a multi-module exercise spanning Module 17 to 20

Please keep these handy while you study these modules. We shall frequently refer to those



Module Outline

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

- Input and Output Specification
- Extraction of Collaborators
- Extraction of Modules
- Extraction of Hierarchy



Input and Output Specification

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Input

- Classes
- Attributes
- Responsibilities

Output

- Collaboration
- Modularization
- Hierarchy



Steps

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

- We analyzed the responsibilities of classes already
- Next the relationships and hierarchy between the classes must be clarified
- Three steps are:
 - Identify possible relationships / collaboration between classes
 - For those with relationships, describe the nature of the relationship
 - Identify the Hierarchy between related classes



Resultant Classes: RECAP (Module 17)

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

- ① Company
- ② Employee
- ③ Executive
- ④ Lead
- ⑤ Manager
- ⑥ Leave
- ⑦ SL
- ⑧ ML
- ⑨ PL
- ⑩ CL
- ⑪ EL
- ⑫ DL
- ⑬ LWP
- ⑭ UL
- ⑮ SysAdmin



Attributes: RECAP (Module 17)

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

- Login ID
- Name
- Employee Code
- Personal details
- Designation
- Salary
- ...



Employee Classes with its Attributes and Responsibilities: RECAP (Module 18)

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Employee	
Attributes	Responsibilities
<ul style="list-style-type: none">• Name• Personal details• Designation• Employee code• Login id	<ul style="list-style-type: none">• Record Daily Attendance• Request for Leave• Cancel an Approved Leave not yet availed• Avail Leave (if approved)• Check / Export own Leave Status for a period

- Similarly attributes and responsibilities can be for Executive, Lead and Manager classes
- ...



Leave Classes with its Attributes and Responsibilities: RECAP (Module 18)

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Leave	
Attributes	Responsibilities
<ul style="list-style-type: none">• Type of Leave• Start Date• Duration of Leave• Employee Id	<ul style="list-style-type: none">• Validity Check• Accounting• •

- Similarly attributes and responsibilities can be for CL, EL, ML ...etc.
- ...



Quality Check: RECAP (Module 18)

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

- The Metrics at this Stage

- ① **Coupling:** Low (was Low)
- ② **Cohesion:** Moderate (was Moderate)
- ③ **Sufficiency:** **Very low** (was Unclear)
- ④ **Completeness:** **Low** (was Very low)
- ⑤ **Primitiveness:** Very low (was Very low)

- Actions required

- Explore Association



Collaboration

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Class Name	Responsibilities	Collaborators
Company	<ul style="list-style-type: none">• Manage	<ul style="list-style-type: none">• Employee• Leave
Employee	<ul style="list-style-type: none">• Record Daily Attendance• Request for Leave	<ul style="list-style-type: none">• Leave
	...	
Executive	<ul style="list-style-type: none">• Record Daily Attendance• Request for Leave	<ul style="list-style-type: none">• Leave• Lead
	...	
Lead	<ul style="list-style-type: none">• Approve Leave• Request for Leave	<ul style="list-style-type: none">• Leave• Executive• Manager
	...	
Manager	<ul style="list-style-type: none">• Credit, Debit, Adjust Leave• Hire, Fire Employee• Force-execute	<ul style="list-style-type: none">• Leave• Executive• Lead• SysAdmin
	...	
Leave	<ul style="list-style-type: none">• Validity Check• Accounting	<ul style="list-style-type: none">• Employee



Collaboration

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy
Employee
Hierarchy

Summary

Class Name	Responsibilities	Collaborators
CL	<ul style="list-style-type: none">• Credit, Prorate• Count, Club	<ul style="list-style-type: none">• Employee• Leave
SL	<ul style="list-style-type: none">• Credit, Carry Over• Prorate, Club	<ul style="list-style-type: none">• Employee• Leave
EL	<ul style="list-style-type: none">• Credit, Carry Over• Accumulate, Encash	<ul style="list-style-type: none">• Employee• Leave
DL	<ul style="list-style-type: none">• Create	<ul style="list-style-type: none">• Employee• Leave
PL	<ul style="list-style-type: none">• Credit, Prorate• Count, Club	<ul style="list-style-type: none">• Employee• Leave
ML	<ul style="list-style-type: none">• Count, Club	<ul style="list-style-type: none">• Employee• Leave
LWP	<ul style="list-style-type: none">• Avail, Club• ...	<ul style="list-style-type: none">• Employee• Leave



Modularization

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Depending upon the collaboration of the classes we can decompose the system into 3 cohesive and loosely coupled modules





Quality Measure

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy
Employee
Hierarchy

Summary

Module	Impact On				
	Coupling	Cohesion	Sufficiency	Completeness	Primitiveness
Company	Low	High	Low	Low	Low
Employee	Low	High	Low	Low	Low
Leave	Low	High	Low	Low	Low



Employee Hierarchy

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy
Employee
Hierarchy

Summary

Class: Employee
Attributes:
<ul style="list-style-type: none">• Name• ID• DoB
Responsibilities:
<ul style="list-style-type: none">• Record Daily Attendance• Request for Leave• Cancel an Approved Leave• Avail Leave, if approved

IS-A

Class: Executive	Class: Lead	Class: Manager
Attributes: <ul style="list-style-type: none">• Reporting_Lead• ...	Attributes: <ul style="list-style-type: none">• Reporting_Manager• List of Reporting Executives	Attributes: <ul style="list-style-type: none">• List of Reporting Leads• ...
Additional Responsibilities: <ul style="list-style-type: none">• Report to Lead	Additional Responsibilities: <ul style="list-style-type: none">• Approve Leave (Executive)• Regret Leave (Executive)• Revoke Leave (Executive)• Report to Manager• Take Reporting (Executive)	Additional Responsibilities: <ul style="list-style-type: none">• Approve Leave (Lead)• Regret Leave (Lead)• Revoke Leave (Lead)• Take Reporting (Lead)

- For brevity, all characteristics and all responsibilities are not shown
- Suggest refinements to Employee hierarchy



Employee Hierarchy

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

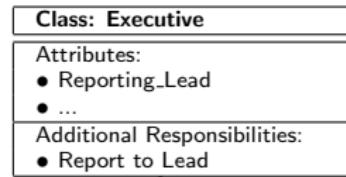
Collaboration

Modularization

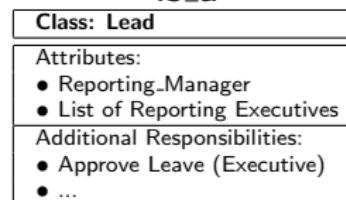
QC

Hierarchy
Employee
Hierarchy

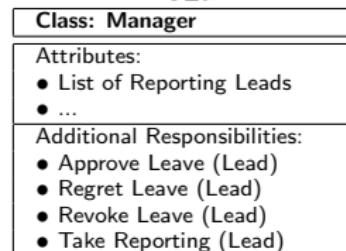
Summary



IS_A



IS_A





Employee Hierarchy

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

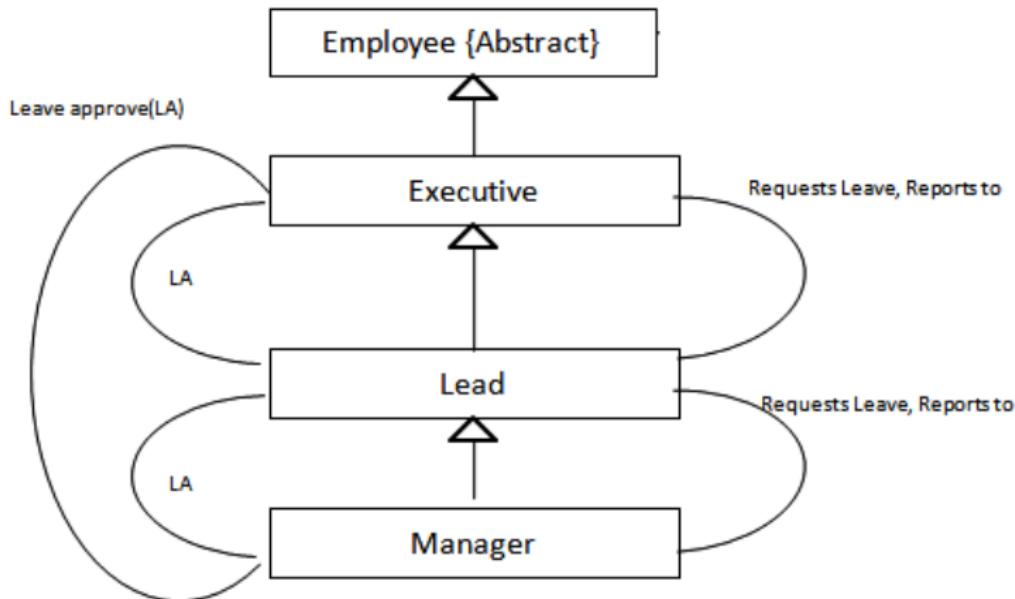
Collaboration

Modularization

QC

Hierarchy
Employee
Hierarchy

Summary





Module Summary

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

- Analyzed the relationships to extract collaboration, modularization and hierarchy



Instructor and TAs

Module 19

Partha Pratim
Das

Objectives &
Outline

Input and
Output
Specification

LMS So Far
QC

Collaboration

Modularization

QC

Hierarchy

Employee
Hierarchy

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

Module 20: Object-Oriented Analysis & Design

Identification of Classes, Objects & Relationships in LMS:

An Exploratory Exercise: Part 4

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

- Perform an exploratory exercise to identify potential classes, objects, and relationships for the Leave Management System (LMS)
- The following will be used as primary source of information for LMS (as received from the client)
 - Written English description of LMS (PDF document as shared)
 - Conversation between client and vendor (Tutorial video as uploaded)
- This will be a multi-module exercise spanning Module 17 to 20

Please keep these handy while you study these modules. We shall frequently refer to those



Module Outline

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

- Refinement of Hierarchy
 - Employee Hierarchy
 - Leave Hierarchy
- Relationships



Employee Hierarchy: RECAP (Module 19)

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

Class: Employee
Attributes: <ul style="list-style-type: none">• Name• ID• DoB
Responsibilities: <ul style="list-style-type: none">• Record Daily Attendance• Request for Leave• Cancel an Approved Leave• Avail Leave, if approved

IS-A

Class: Executive	Class: Lead	Class: Manager
Attributes: <ul style="list-style-type: none">• Reporting_Lead• ...	Attributes: <ul style="list-style-type: none">• Reporting_Manager• List of Reporting Executives	Attributes: <ul style="list-style-type: none">• List of Reporting Leads• ...
Additional Responsibilities: <ul style="list-style-type: none">• Report to Lead	Additional Responsibilities: <ul style="list-style-type: none">• Approve Leave (Executive)• Regret Leave (Executive)• Revoke Leave (Executive)• Report to Manager• Take Reporting (Executive)	Additional Responsibilities: <ul style="list-style-type: none">• Approve Leave (Lead)• Regret Leave (Lead)• Revoke Leave (Lead)• Take Reporting (Lead)

- For brevity, all characteristics and all responsibilities are not shown
- Suggest refinements to Employee hierarchy



Employee Hierarchy: RECAP (Module 19)

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

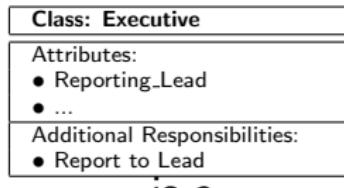
Employee
Hierarchy

Leave
Hierarchy

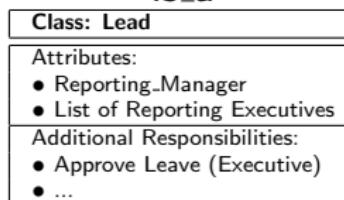
QC

Relationship

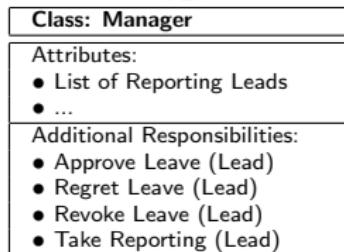
Summary



IS_A



IS_A





Employee Hierarchy: RECAP (Module 19)

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

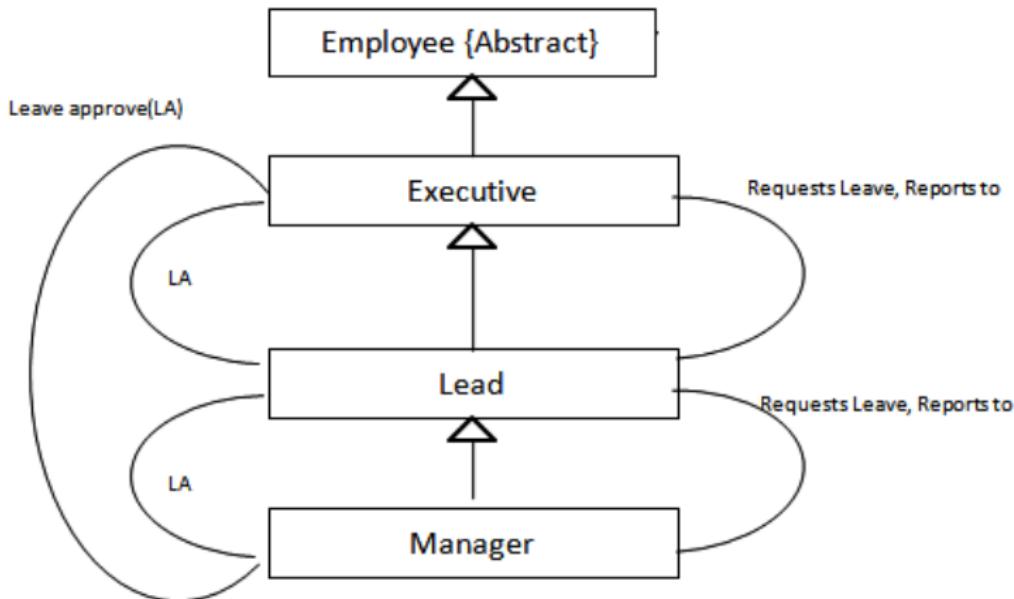
Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary





Leave Hierarchy

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

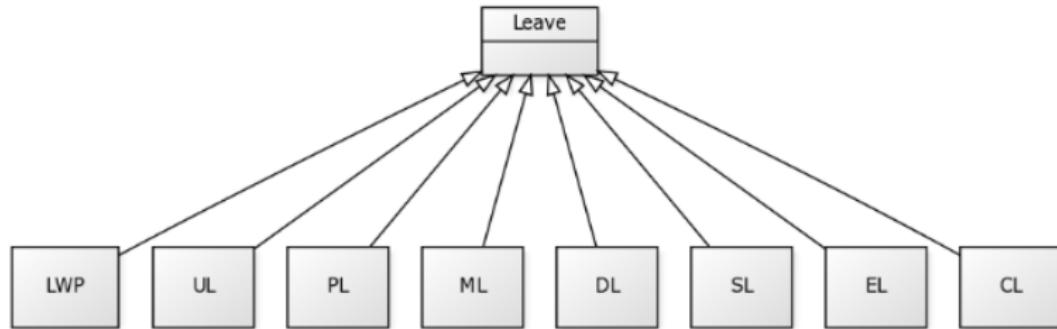
Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary



• Quality Check

- Hierarchy is rather Wide and Shallow
- Action: Identify Intangible Concepts to Balance



Leave Hierarchy: Analysis of Pre– and Post– Conditions

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

Property	CL	EL	SL	DL	ML	PL	LWP	UL
<i>Entitlement</i>	Y	Y	Y	NA	Y ^a	Y	Y	N ^b
<i>Duration of Leave</i>	Y	Y	Y	NA	Y	Y	Y	N ^c
<i>Is Leave Clubbable?</i>	N	Y	Y	NA	Y	Y	Y	N
<i>Is Holiday exempt in Leave?</i>	Y	N	N	NA	N	N	N	N
<i>Must Leave be Pre-Approved?</i>	N	Y	N ^d	NA	Y	N ^e	Y	N
<i>Does Leave Carry-over & Accumulate?</i>	N	Y	Y	NA	N	N	N	N
<i>Can Leave be Encashed?</i>	N	Y	N	NA	N	N	N	N
<i>Does Leave need Certification?</i>	N	N	Y	NA	Y	Y	N	N
<i>Is Leave paid?</i>	Y	Y	Y	NA	Y	Y	N	N

^a: Only for female, when pregnant, twice in career

^b: Deemed entitlement for a week before actions start

^c: Allowed for up to a 7 days

^d: Exception condition for sickness

^e: Exception condition for parenthood



Leave Hierarchy

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

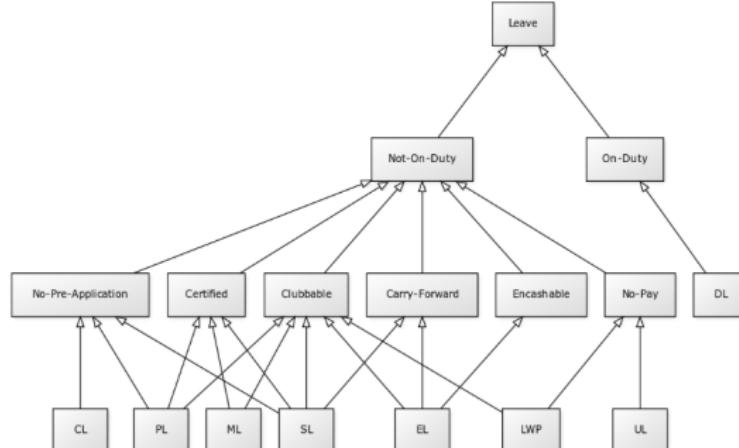
Leave
Hierarchy

QC

Relationship

Summary

Property	CL	EL	SL	DL	ML	PL	LWP	UL
<i>Entitlement</i>	Y	Y	Y	NA	Y ^a	Y	Y	N ^b
<i>Duration of Leave</i>	Y	Y	Y	NA	Y	Y	Y	N ^c
<i>Is Leave Clubbable?</i>	N	Y	Y	NA	Y	Y	Y	N
<i>Is Holiday exempt in Leave?</i>	Y	N	N	NA	N	N	N	N
<i>Must Leave be Pre-Approved?</i>	N	Y	N ^d	NA	Y	N ^e	Y	N
<i>Does Leave Carry-over & Accumulate?</i>	N	Y	Y	NA	N	N	N	N
<i>Can Leave by Encashed?</i>	N	Y	N	NA	N	N	N	N
<i>Does Leave need Certification?</i>	N	N	Y	NA	Y	Y	N	N
<i>Is Leave paid?</i>	Y	Y	Y	NA	Y	Y	N	N





Leave Hierarchy

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

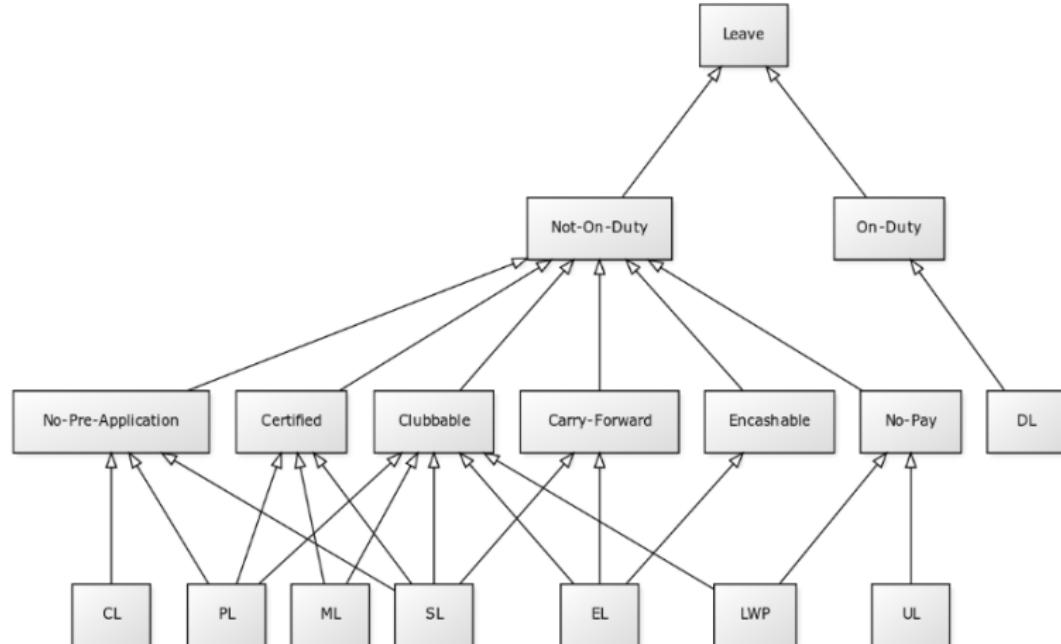
Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary



- **Quality Check**

- **Balanced Hierarchy**



Quality Measure

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

Module	Impact On				
	Coupling	Cohesion	Sufficiency	Completeness	Primitiveness
Company	Low	High	Low	Low	Low
Employee	Moderate	High	Low	Moderate	Moderate
Leave	Moderate	High	Low	Moderate	Moderate



Relationship among Employee Classes

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

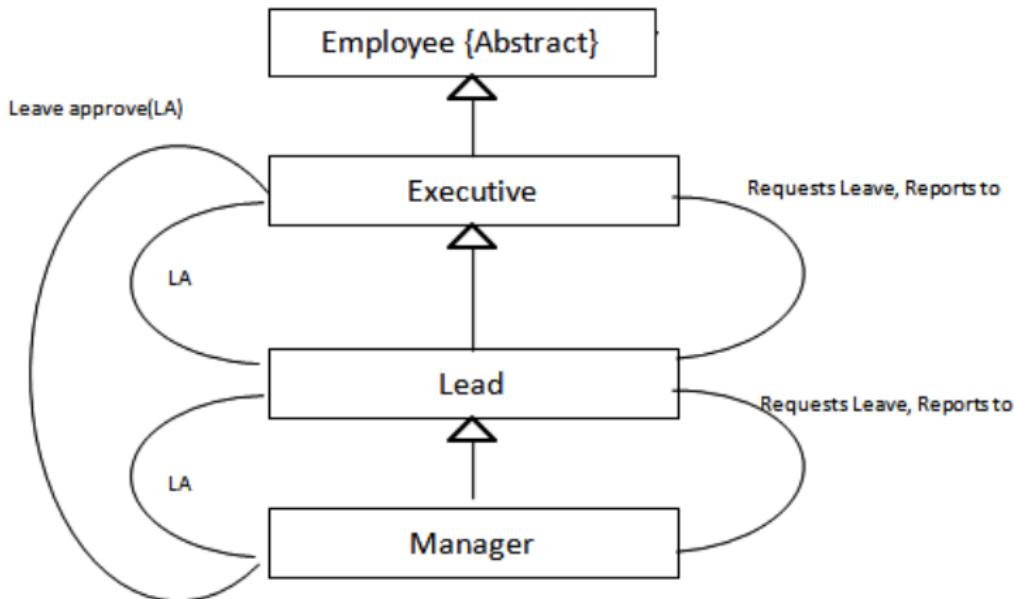
Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary





Relationship among Classes

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

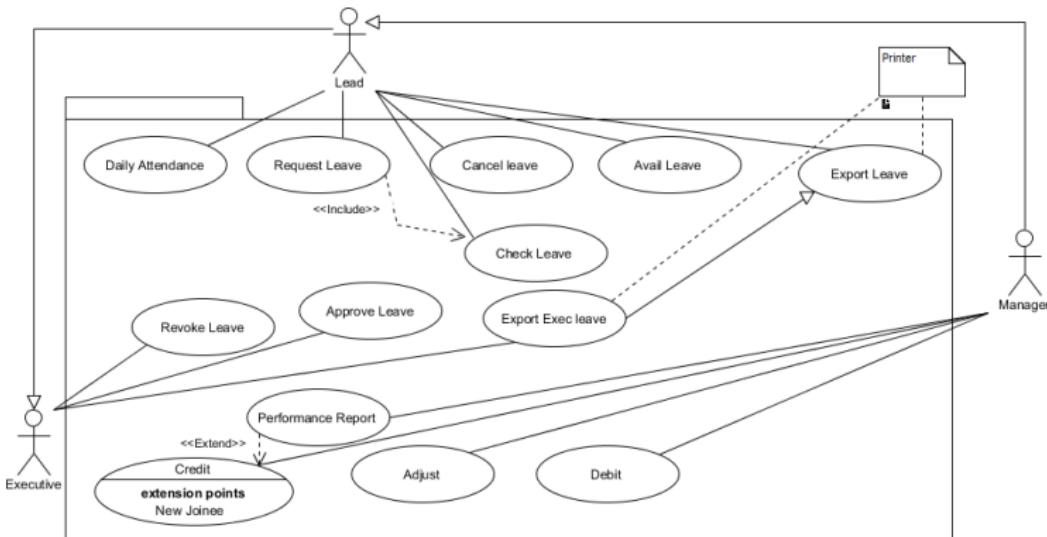
Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary



Employees and Leave Actions are Related



Module Summary

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

- Understood the refinement of hierarchy
- Outlined relationships
- Need to have tools for better capture and refinement – UML to be used to refine the design further



Instructor and TAs

Module 20

Partha Pratim
Das

Objectives &
Outline

Hierarchy

Employee
Hierarchy

Leave
Hierarchy

QC

Relationship

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

Module 21: Object Oriented Analysis & Design

Overview of UML

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?
What is the
UML *not*?
History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams
Features of
Structural
Diagrams

Summary

- Discuss the overview of Unified Modeling Language (UML)



Module Outline

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?
What is the
UML *not*?
History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams
Features of
Structural
Diagrams

Summary

● Overview of UML

- Why UML?
- What is the UML?
- What is UML is not?
- History of UML

● UML Diagrams

- Diagram Classification
- Features of Structural Diagrams
- Features of Behavioral Diagrams



Why UML?

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

A notation for system analysis and design

- Is a standard
- Helps a designer or developer to capture artifacts of software design
- Eliminates the issues of consistencies and accuracy in software design and analysis
- Minimizes vagueness and imprecision in expression
- Facilitates robust communication within the teams and with client



What is the UML?

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?
What is the
UML *not*?
History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams
Features of
Structural
Diagrams

Summary

UML

- Provides users with an **expressive modeling language**
 - for the **specification, construction, visualization, and documentation** of the artifacts of a software system
 - for the construction of different kinds of models
 - for the exchange of models
- Provides users with **ready-to-use core concepts**
 - however, extensibility and specialization mechanisms are available
- Provides a **formal basis** for understanding the modeling language
 - *meta-model* in terms of a UML class diagram
 - *Semantics* is part of the official UML documentation
- Supports **higher-level development concepts**
 - such as collaborations, patterns, and components
- Integrates **Best Practices**



What is the UML *not*?

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

UML does not prescribe

- a certain **process**
- a certain **modeling tool**
- any **modeling guidelines**
- a certain **programming language**

Dedicated goal of UML is: **OPENNESS**



How UML was formed?

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- The Unified Modeling Language (UML) is the primary modeling language used to analyze, specify, and design software systems
- From the late 1980s and well into the 1990s, numerous methodologies arose and were subsequently modified and refined. Many of these were strong in certain areas, weak in others
- In the mid-1990s, Booch, Rumbaugh, and Jacobson joined forces at Rational Software Corporation and began to meld their respective methodologies to create what would be the first version of the UML
- In November 1997 the Object Management Group (OMG) adopted the UML as a standard



How UML was formed?

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

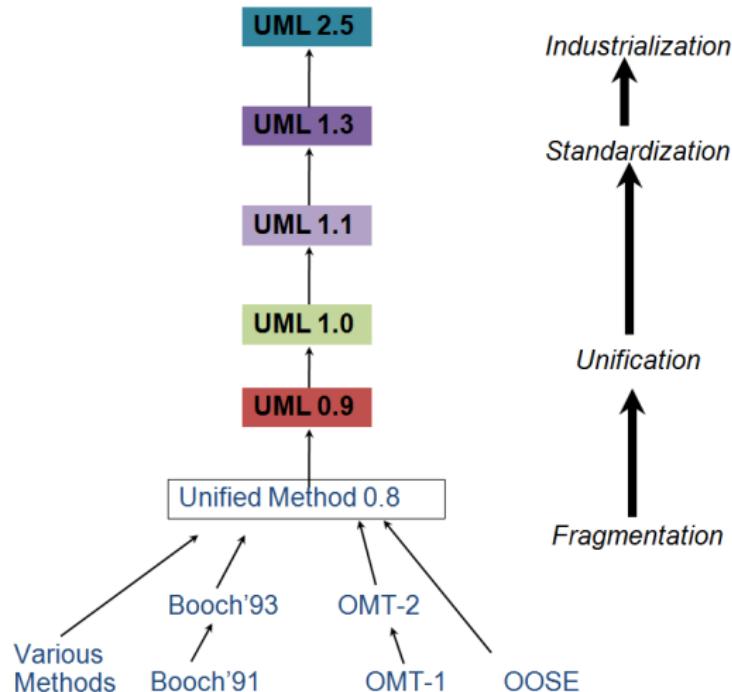
UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary





Concept of Diagrams

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- UML consists of various diagrams to capture the various aspects of software design
- A system may contain numerous classes and many inter relationships, so we can group the class diagrams logically depicting one aspect of the system. Creating a singular class diagram is not the solution
- Across all diagrams, all components with the same name are considered to be references to the same model item. A component may appear in multiple diagrams



Diagram Classification: UML 2.5

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

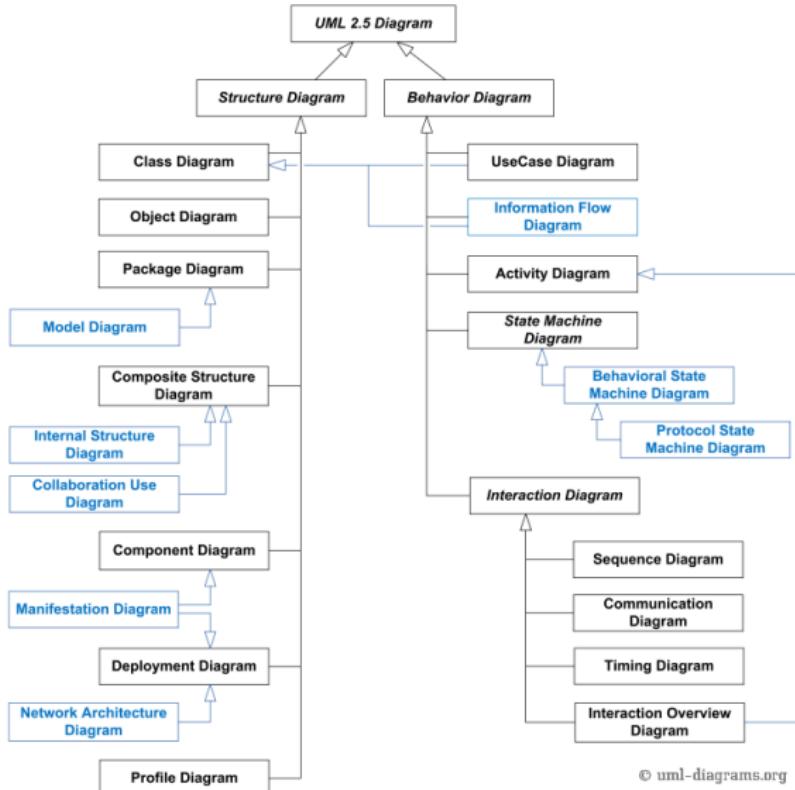
UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary



© uml-diagrams.org



Diagram Classification: UML 2.5

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- **Structural Diagrams** show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other
 - The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts
- **Behavioral Diagrams** show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Behavioral Diagrams

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- **Use case diagrams** represents the requirements of a system including internal and external influences. This diagram is used to capture the user specifications for a system to be built
- **Activity diagrams** models the flow from one activity to another activity. The activity can be described as an operation of the system. It helps to capture the system behavior in the context of activities
- **State-Chart diagrams** are used to describe the different states of an object during its life time. It is used to capture the states to model the lifetime of a system



Behavioral Diagrams: Interaction Diagrams

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- **Sequence diagrams** captures the temporal ordering of messages exchanged between objects during their lifetime respectively. It is used to model the system behavior
- **Communication diagrams** captures the spatial ordering of messages exchanged between objects during their lifetime respectively. It is used to model the system behavior
- **Timing diagrams** captures the change of states of an element or elements change over time and how events change those states. It is used to model real time system behavior
- **Interaction Overview diagrams** models the total interaction flow consisting from various sequence, collaboration and activity flows. It provides an overall flow control among various interactions in the system



Structural Diagrams

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- **Class diagrams** models the analysis and design of the static view of an application. It captures the key abstractions of the system at the first level, and is later refined with details of every abstraction for reference
- **Object diagrams** represents an instance of a class diagram. It is used to capture the static view of a system at a particular moment
- **Package diagrams** encapsulates the major components of a system and highlights the dependencies. It is used to capture the dependencies between the major components of a system



Structural Diagrams

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- **Composite Structure diagrams** describes the complete structure of a system along with its sub-parts and interfaces. It is used to capture internal structure view of a system
- **Component diagrams** models physical aspects of a system. It is used to capture the static implementation view of a system consisting of the components and their relationships
- **Deployment diagrams** describes the hardware components where software components are deployed. It is used to capture the hardware topology of a system
- **Profile diagrams** allows to define custom stereotypes, tagged values, and constraints as a lightweight extension mechanism to the UML standard. Profiles allow to adapt the UML meta-model for different platforms and domains



Module Summary

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?

What is the
UML *not*?

History of UML

UML
Diagrams

Diagram
Classification

Features of
Behavioral
Diagrams

Features of
Structural
Diagrams

Summary

- UML is required to create a standard notation for designing and modeling software systems
- UML majorly contains two types of diagrams, Structure and Behavior to capture a view of the system
- The overall system view is captured by collection of all the required diagrams capturing a certain view of the system



Instructor and TAs

Module 21

Partha Pratim
Das

Objectives &
Outline

Overview of
UML

Why UML?
What is the
UML?
What is the
UML *not*?
History of UML

UML
Diagrams

Diagram
Classification
Features of
Behavioral
Diagrams
Features of
Structural
Diagrams

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase
Analysis Phase
Design Phase
Implementation
Phases

Summary

Module 22: Object Oriented Analysis & Design

SDLC Phases and UML Diagrams

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase

Design Phase

Implementation
Phases

Summary

- Understand Software Development Life Cycle (SDLC)
- Understand how UML diagrams contribute to the SDLC phases?



Module Outline

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase

Design Phase

Implementation
Phases

Summary

- SDLC phases
- Phase-wise Diagrams
 - Requirements Specification Phase
 - Analysis Phase
 - Design Phase
 - Implementation Phases



What is a Software Development Life Cycle?

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase
Design Phase
Implementation
Phases

Summary

- SDLC is a process followed for a software project, within a software organization
- It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software
- The life cycle defines a methodology for improving the quality of software and the overall development process



Phases of an SDLC

Module 22

Partha Pratim
Das

Objectives &
Outline

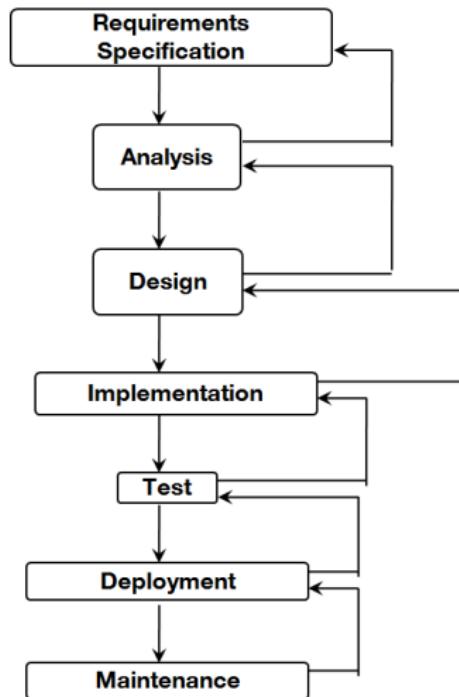
SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase
Design Phase
Implementation
Phases

Summary





OOAD in SDLC

Module 22

Partha Pratim
Das

Objectives &
Outline

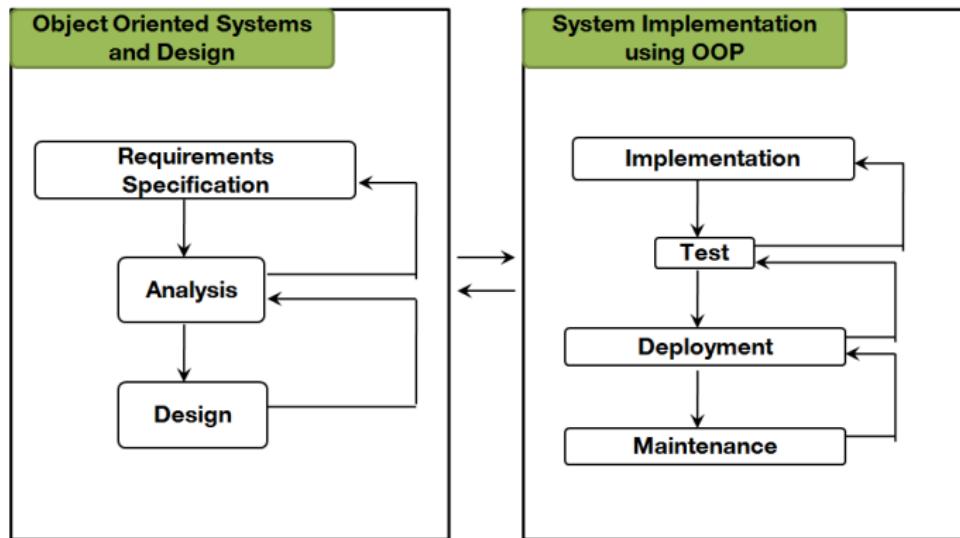
SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase
Design Phase
Implementation
Phases

Summary





Models of the Requirements Specification Phase

Module 22

Partha Pratim
Das

Objectives &
Outline

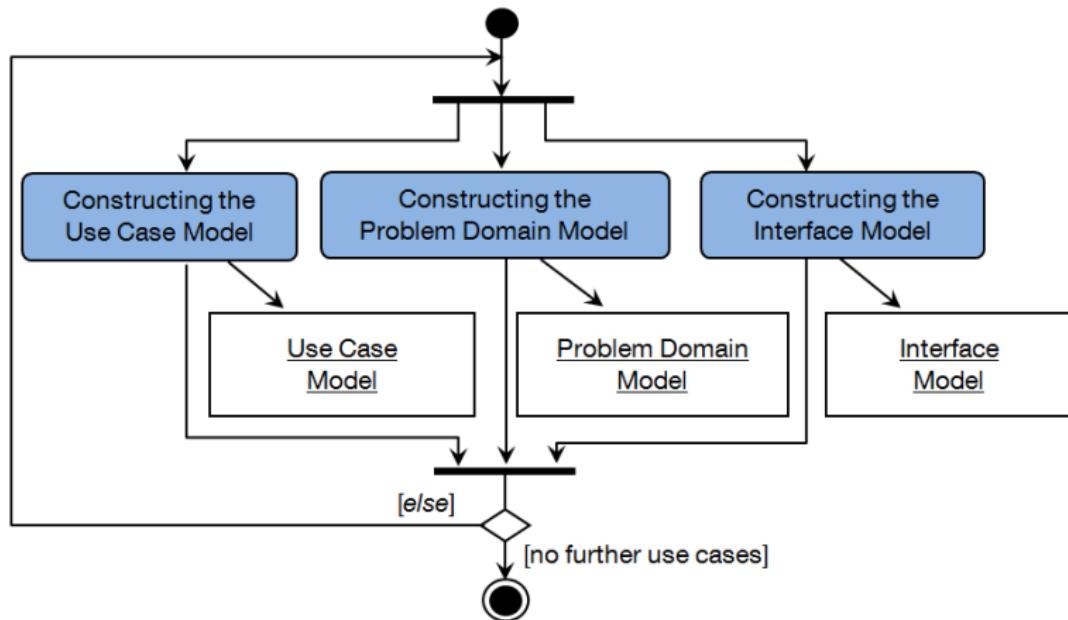
SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase
Design Phase
Implementation
Phases

Summary





Output of the Requirements Specification Phase

Module 22

Partha Pratim
Das

Objectives &
Outline

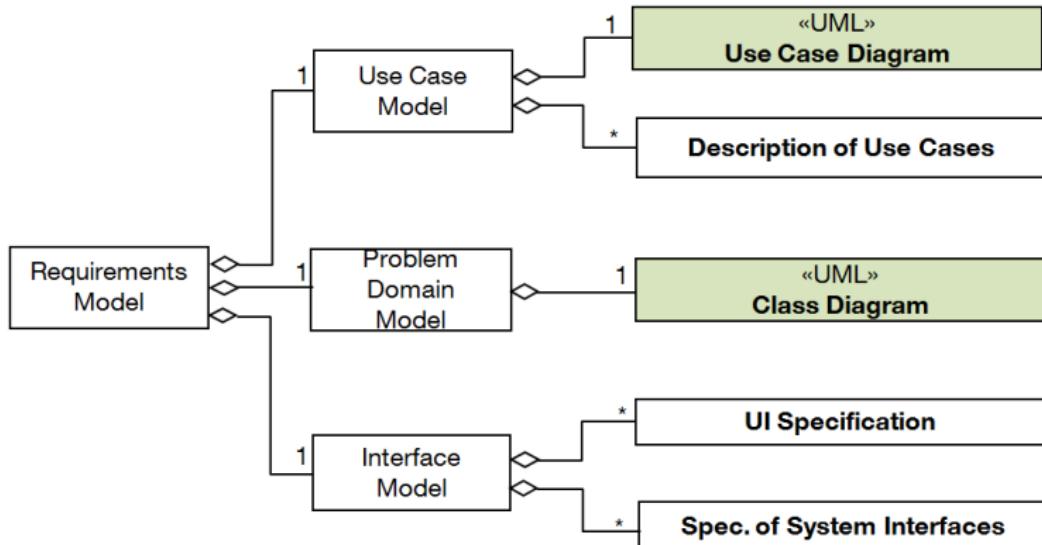
SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase
Design Phase
Implementation
Phases

Summary





Models of the Analysis Phase

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

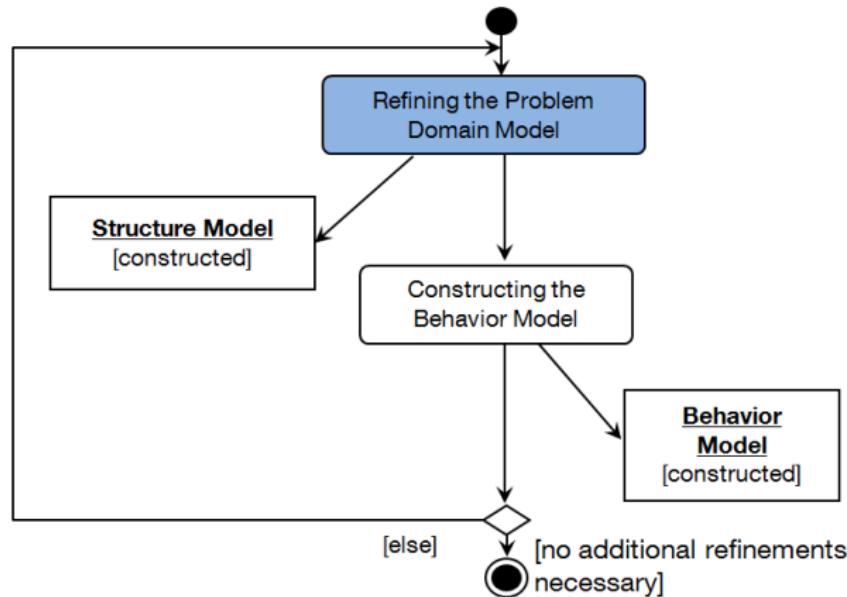
Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase

Design Phase
Implementation
Phases

Summary





Output of the Analysis Phase

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

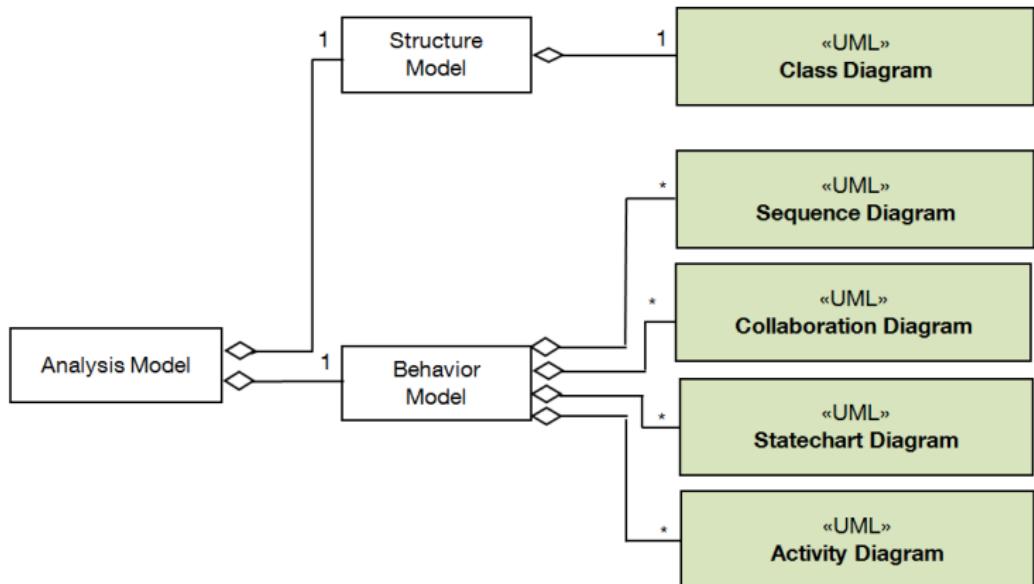
Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase

Design Phase
Implementation
Phases

Summary





Models of the Design Phase

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

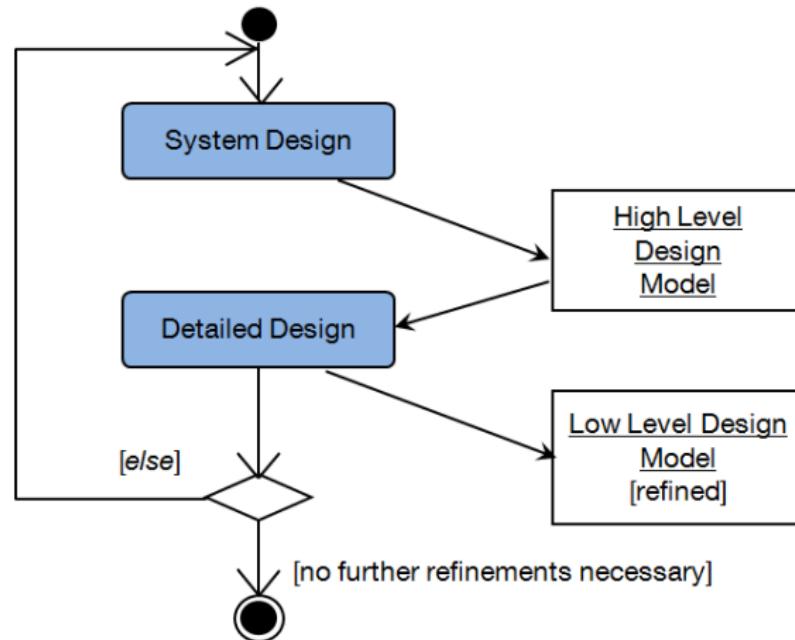
Requirements
Specification
Phase

Analysis Phase

Design Phase

Implementation
Phases

Summary





Output of the Design Phase

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

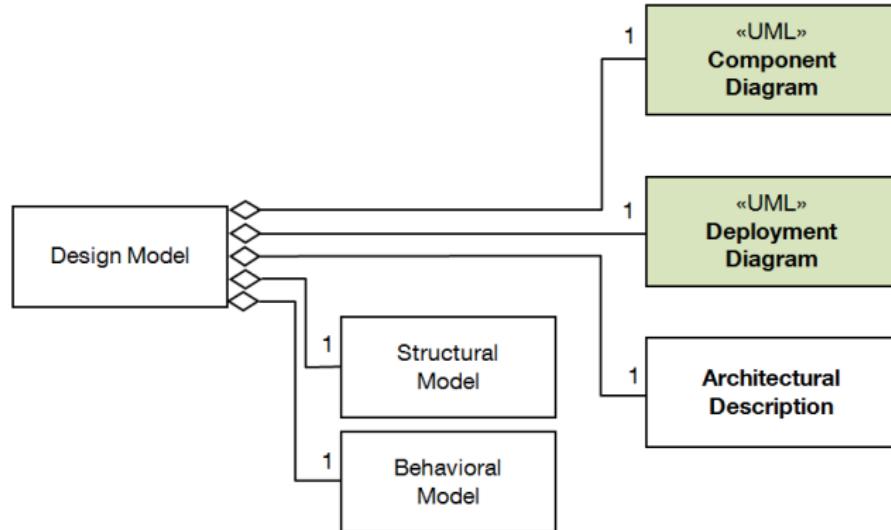
Requirements
Specification
Phase

Analysis Phase

Design Phase

Implementation
Phases

Summary





Phases for System Implementation

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase
Design Phase

Implementation
Phases

Summary

- The other phases of SDLC

- Implementation
- Test
- Deployment
- Maintenance

implement the models defined in earlier phases

- The Diagrams are used to implement the respective components in the implementation
- These are implemented using various OOP languages, and some models continue to be used



Module Summary

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase

Design Phase

Implementation
Phases

Summary

- SDLC phases are understood for the development process
- Role of various UML diagram in different phases are discussed



Instructor and TAs

Module 22

Partha Pratim
Das

Objectives &
Outline

SDLC Phases

Phase-wise
Diagrams

Requirements
Specification
Phase

Analysis Phase

Design Phase

Implementation
Phases

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

Module 23: Object Oriented Analysis & Design

Use-Case Diagrams: Part 1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

● Understanding Use Case Diagrams



Module Outline

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

- Use Case Diagrams
- Actors
- Use Cases
 - Specification of Use-Cases



What are Use case Diagrams

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

- The integration of business knowledge with the development specification is a requirement
- The development organization knows the specifications for developing a module, but does not know who will interact with these modules and for what purpose
- Use Case Diagrams depicts the human interaction with the system to give the context of who uses, which part of the system and for what purpose



What are Use case Diagrams

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

- Use case diagrams are behavior diagrams used to describe a set of actions (use cases) that systems perform in collaboration with one or more external users of the system (actors)
- Use case diagrams are mainly used to gather requirements of a system and identify the internal and external factors influencing the system.



Components of Use case Diagrams

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

The use case diagram is composed of

- Actors
- Use cases and their specifications
- Relationships between Use cases



Actor

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

- Actors are entities that interface with the system
- They can be people or other systems
- Actors are depicted as stylized stick figures
- This stick figures are used as stereotypes to depict many models at the same time like process, thread, meta-class, power-type etc labeled with guillemets <>>
- The roles the actors play in the system is important, not their real world identity



Actor

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

Actors can be classified as

- **human**: e.g. novice/trained user; system administrator
- **non-human**: e.g., fax, e-mail
- **primary**: ultimate user of the system
- **secondary**: ensures the correct functionality of the system
- **active**: initiates use cases
- **passive**: corresponding use case is initiated by the system



Actor

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

How can Actors be identified?

- Who uses the essential use cases?
- Who needs system support in order to fulfill the daily tasks?
- Who is responsible for system administration?
- What are the external devices/software systems the system has to communicate with?
- Who is interested in the results of the system?



LMS: List of all Identified Nouns: RECAP (Module 17)

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			



Actors in LMS

Module 23

Partha Pratim
Das

Objectives &
Outline

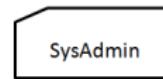
Use Case
Diagrams

Actors

Use-Cases
Specification of
Use-Cases

Summary

- In the LMS, the human actors are Manager, Lead and Executive.
- The non human actors is the printer
- A Secondary actor is the SysAdmin





Use Cases

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

- Use cases represent what the actors want your system to do for them
- Each use case is a complete course of events to be executed in the system from a user's perspective
- Use cases can contain short descriptions course of events in the system from a user's perspective



LMS: List of all Identified Verbs: RECAP (Module 18)

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

Wants	Manage	Work	Report
Approve	Regret	Credit	Join
Prorate	Cross	En-cash	Paid
Allow	Send	Need	Become
Enjoy	Avail	Proceeding	Employ
Consider	Deduct	Provide	Request
Cancel	Check	Export	Revoke
Debit	Adjust	Perform	Hire
Fire	Generate	Leave	Can be Availed
Can be	Can't be	Can't be	Can't be
Clubbed	Availed	Carried forward	Clubbed
Can't be	Accumulated	Proposed for	Join Back
Continued	Up		
Doesn't Draw	Can be Revoked	Leave Credited	

Many extracted verbs are in derived forms – so we extract the unique stems



Use Cases in LMS

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

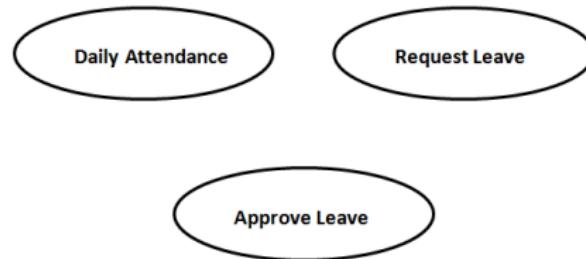
Actors

Use-Cases

Specification of
Use-Cases

Summary

- Shown below are important use cases executed by actors of Leave Management System. **Request Leave**, **Daily Attendance** are use cases, that is, functionality which all the three human actors need from the system. Similarly approve leave use case executed by only Lead and Manager





Specification of Use Case

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

- A use case can be specified in the following manner. Depicted below for Request Leave.
 - **Use Case Name:** Request Leave
 - **Use Case Purpose:** The Executive, Lead and Manager, all of them uses this functionality to request for leaves in the system.
 - **Use Case Pre-condition:** User has login id and password to enter the system
 - **Use Case Post-condition:** A new leave request is entered in the system for the user
 - **Failure Conditions:** User does not have valid credentials to enter the system, the user do not have sufficient leave balance
 - **Actors:** Lead, Executive, Manager
 - **Optimistic Flow:**
 - The Executive, Lead and Manager checks for the leave balance.
 - If leave balances are available, leave request added.



Module Summary

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

- Use case diagrams help to integrate business knowledge with the development specification
- Use Case diagram is used to model the various behaviors (usecases) of a system, and the external elements using and executing them (actors)



Instructor and TAs

Module 23

Partha Pratim
Das

Objectives &
Outline

Use Case
Diagrams

Actors

Use-Cases

Specification of
Use-Cases

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases
Include
Extend
Generalization

Summary

Module 24: Object Oriented Analysis & Design

Use-Case Diagrams: Part 2

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- Understanding the relationships among Use Cases



Module Outline

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- Relationship among Use-Cases
 - Includes
 - Extends
 - Generalization



Relationships among Use-Cases

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- Use-Cases share various kinds of relationships
- A relationship between two Use-Cases is basically a dependency between the two Use-Cases
- Defining the relationship between two Use-Cases is the decision of the modeler of the use case diagram
- We discuss the following three relationships among Use-Cases
 - <<include>>
 - <<extend>>
 - Generalization



Relationship among Use-Cases: <<include>>

Module 24

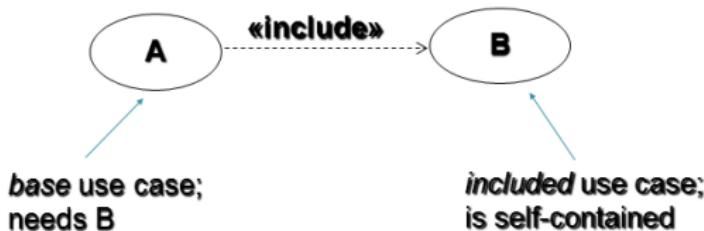
Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases
Include
Extend
Generalization

Summary

- The <<includes>> relationship involves one Use-Case **including** the behavior of another Use-Case in its sequence of events and actions
- Thus, the includes relationship explores the issue of reuse by factoring out the commonality across Use-Cases
- <<includes>> relationship:



- the behavior of B is included into A
- the included use case B is necessary to ensure the functionality of the base use case A



Relationship among Use-Cases: <<include>>

Module 24

Partha Pratim
Das

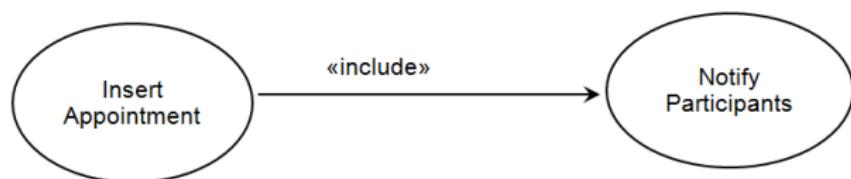
Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- In a calendar, if an appointment is inserted by *Insert Appointment*, the participants will be notified of appointment by included Use-Case *Notify Participants*
- The included Use-Case *Notify Participants* is necessary to ensure the functionality of the base Use-Case *Insert Appointment*



- The Use-Case *Notify Participants* may be included in other Use-Cases – for example, in *Cancel Appointment*



<<include>> in LMS

Module 24

Partha Pratim
Das

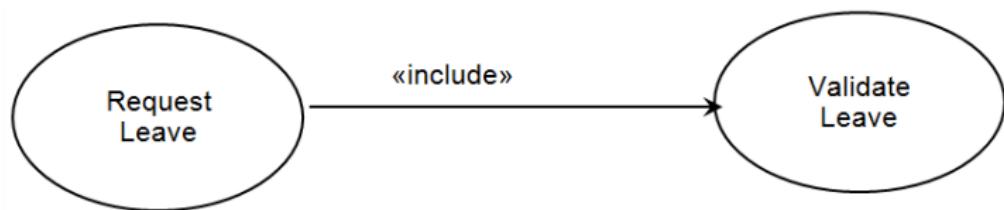
Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- *Validate Leave* Use-Case is included in the Use-Case *Request Leave*
- The include Use-Cases *Validate Leave* is necessary to complete the functionality of the base *Request Leave*



- The Use-Case *Validate Leave* will be included in *Approve Leave* too – and so on



Relationship among Use-Cases: <<extend>>

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- The <<extend>> relationship among the Use-Cases is used to show optional system behavior
- An optional system behavior is extended only under certain conditions, known as **Extension Points**



Relationship among Use-Cases: <<extend>>

Module 24

Partha Pratim
Das

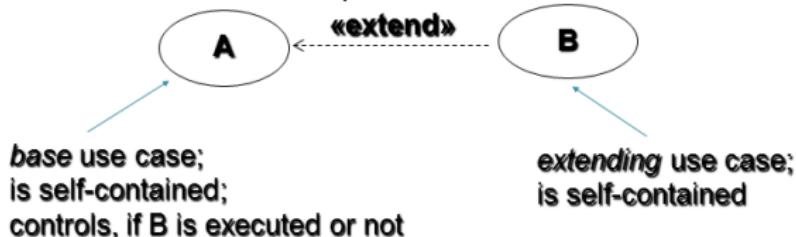
Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- <<extend>> relationship:



- the behavior of B may be incorporated into A
- the extending Use-Case B may be (but need not be) activated by the base Use-Case A
- **extension points** specify the location where the extending Use-Cases extends the base Use-Case
- the **condition** under which the extending Use-Case is incorporated has to be specified
- more than one extension point can be specified for each Use-Case
- the names of extension points have to be unique
- the names of extension points need not be equal with the names of the extending Use-Cases



Relationship among Use-Cases: <<extend>>

Module 24

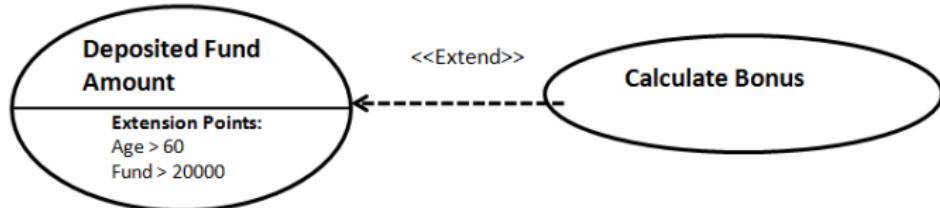
Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases
Include
Extend
Generalization

Summary

- In a savings bank account, bonus is provided only if the deposited fund is above 20,000 or the depositor is above the age of 60 years
- The behavior of *Calculate Bonus* may be incorporated into *Deposited Fund amount*
- At **extension point**: Age above 60 years, Deposit > 20,000





<<extend>> in LMS

Module 24

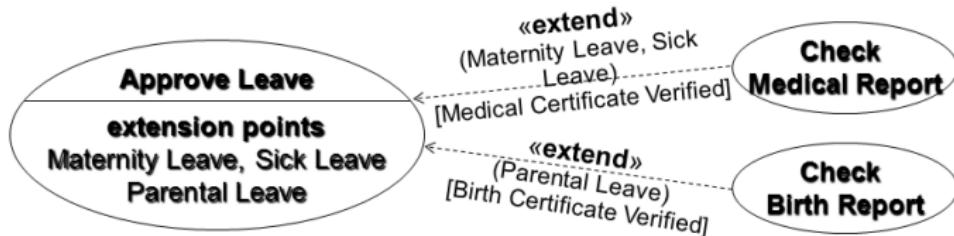
Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases
Include
Extend
Generalization

Summary

- The behavior of *Check Report* may be incorporated into *Approve Leave*
- The extending Use-Case may be (but need not be) activated by the base Use-Case *Approve Leave* at extension points: Medical Leave, Maternity Leave
- Extending Use-Case (*Check Report*) extends the base Use-Case (*Approve Leave*)





Relationship among Use-Cases: Generalization – Concept of Hierarchy

Module 24

Partha Pratim
Das

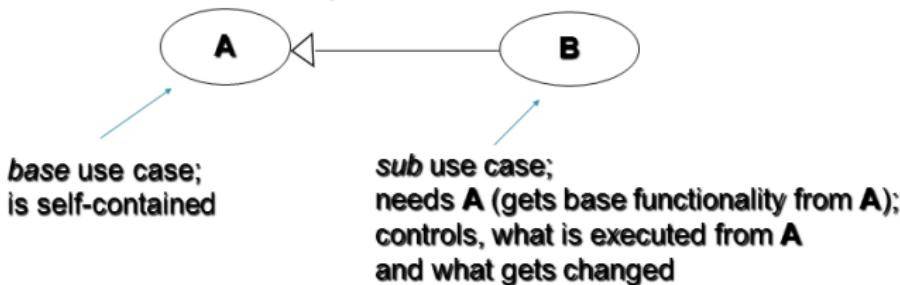
Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- Generalization works the same way with Use-Cases as it does with classes
- The child Use-Case inherits the behavior and meaning of the parent Use-Case
- Generalization helps us to depict the hierarchy present between Use-Cases
- Generalization relationship:



- Similar to the generalization relationship between classes
- B **inherits** the behavior of A and is allowed to **override** and **extend** it
- B **inherits** all relationships of A
- Modeling of **abstract Use-Cases** is also possible (abstract)



Relationship among Use-Cases: Generalization – Concept of Hierarchy

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- *Authentication by Fingerprint* Use-Case will inherit the base Use-Case *Authentication*, to include the basic algorithm, but it adds on features like finger print matching





Generalization in LMS

Module 24

Partha Pratim
Das

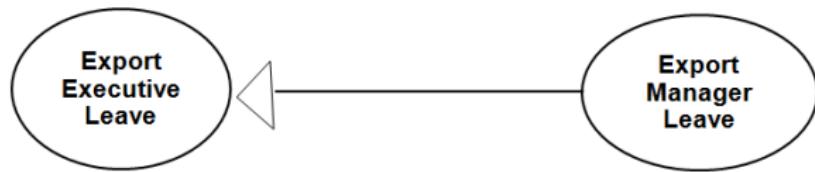
Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- In LMS, *Export Manager Leave* Use-Case inherits the behavior of the parent Use-Case *Export Executive Leave*, that is, it contains the list of leaves but along with it, it contains additional work responsibilities of each of the executives. Hence a specialized Use-Case





Generalization among Actors

Module 24

Partha Pratim
Das

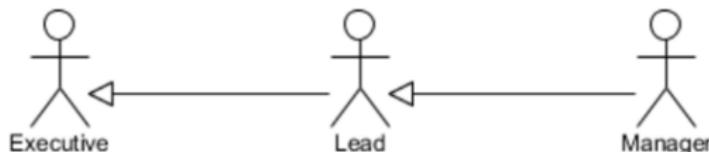
Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- There can be generalization among actors, which can be captured in the use case diagrams
- Generalization among actors, specify that use cases executed by the base actor is inherited by the derived actor
- The derived actor can execute extra use cases



Generalization Among Actors



Module Summary

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

- Relationships among Use-Cases are classified and discussed
- Relationships among Use-Cases are identified for LMS



Instructor and TAs

Module 24

Partha Pratim
Das

Objectives &
Outline

Relationships
among
Use-Cases

Include
Extend
Generalization

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Module 25: Object Oriented Analysis & Design

Use-Case Diagrams: Part 3

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- Exploring the Use-Case Diagrams for Leave Management System (LMS)



Module Outline

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

For the Leave Management System we shall

- Identify Actors
- Identify Use-Cases
- Identify the Relationships among Use-Cases
- Build Use-Case Diagram for LMS



Use-Cases Example: RECAP (Module 24)

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

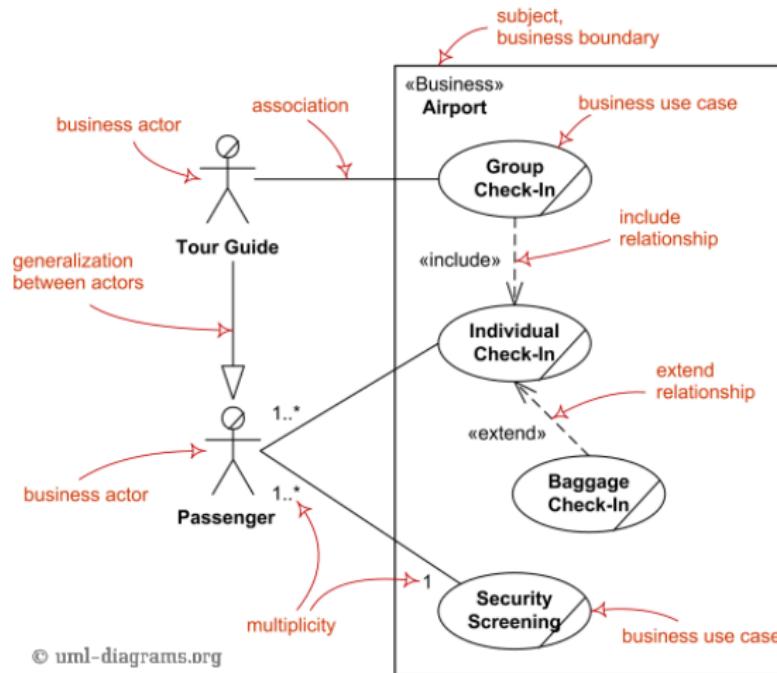
Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Use-Cases Example: RECAP (Module 24)

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

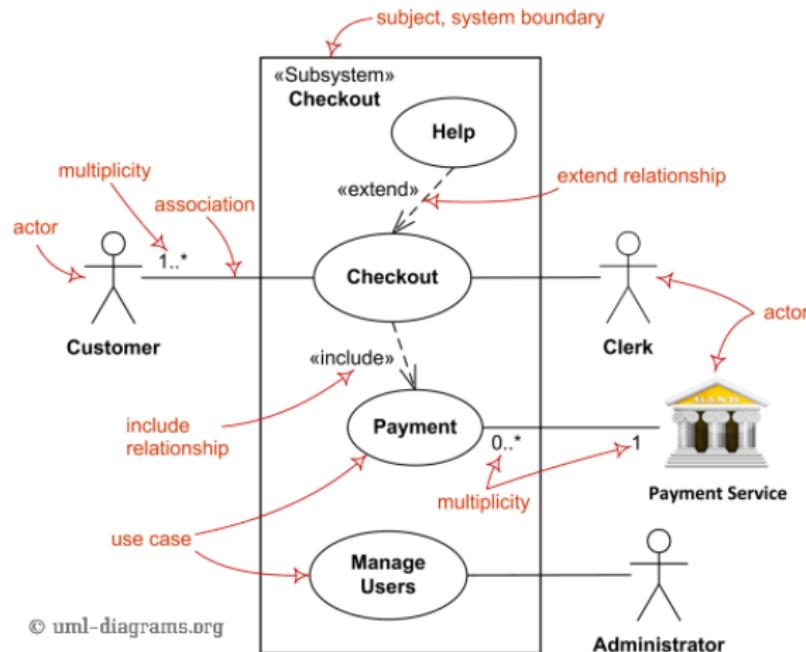
Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Identify Actors

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Identify Actors for LMS



List of all Identified Nouns: RECAP (Module 17)

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Company	Attendance	Leave	Employees
Contributors	Lead	Executive	Manager
Leave Rules	Days	Year	Name
Type of Leave	Period	Absence	Holiday
PL	CL	EL	DL
SL	ML	LWP	UL
Pre-approval	Month	Service	Quarter
Medical Certificate	Parenthood Certificate	Disciplinary Action	Administration Function
Daily Attendance	Personal Details	Calender Year	Batch Task
Account	Balance	Designation	SysAdmin
Parent	Salary	Week	List
Privilege	Right	Login ID	Leave Status
Employee Code			



Actors are Nouns

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- Reading through the specification of the Leave Management System, we identify the actors
- Actors are the nouns in the specifications
 - Executive
 - Lead
 - Manager
 - Printer
- Now based on people or things, we identify the human actors and non-human actors
 - Human
 - Executive
 - Lead
 - Manager
 - Non-Human
 - Printer
 - Secondary
 - SysAdmin



Actors in LMS

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary



Executive



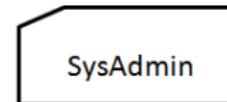
Lead



Manager



Printer



SysAdmin



Identify Use-Cases

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Identify Use-Cases for LMS



List of all Identified Verbs (Stem Only): RECAP (Module 18)

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Wants	Manage	Work	Report
Approve	Regret	Credit	Join
Prorate	Cross	En-cash	Pay
Allow	Send	Need	Become
Enjoy	Avail	Proceed	Employ
Consider	Deduct	Provide	Request
Cancel	Check	Export	Revoke
Debit	Adjust	Perform	Hire
Fire	Generate	Club	Carry forward
Continue	Accumulate	Propose	Join Back
Draw			



Use-Cases are Verbs

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- Reading through the specification, we identify several verbs which are behaviors, that is, Use-Cases of the system

- Daily Attendance
- Request Leave
- Cancel Leave
- Avail Leave
- Export Leave
- Approve Leave
- Revoke Leave
- Export Exec Leave
- Check Medical Report
- Adjust
- Debit
- Credit



Identify Relationships

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Identify Relationships for LMS



Includes

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

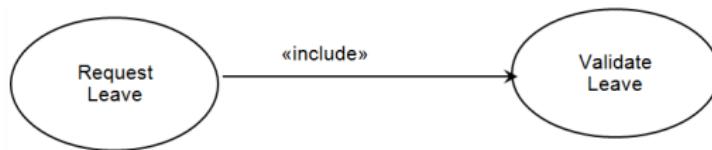
Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- Some verbs will have auxiliaries, that is, related tasks, required to complete its action
- Like *Request Leave* will include the verb *Check Leave*. Hence an includes definition is identified





Extend

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

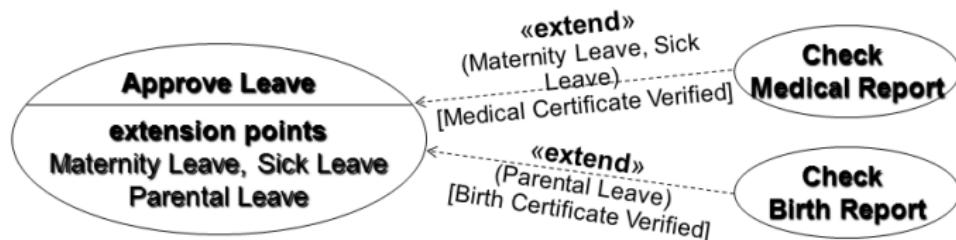
Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- Some verb will incorporate another verb at some conditions, that is, extension points
- In case of Medical or Maternity Leaves, the verb *Approve Leave* will be extended by the action *Check Medical Report*





Generalization among Use-Cases

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- There can be hierarchy among Use-Cases, that is, one of the Use-Cases inherits the behavior of another Use-Case, along with some extra behaviors
- We see *Export Exec Leave* has the same behavior like *Export Leave* (will be executed by all actors), along with some special actions for the Lead and Manager, as only they will be executing the Use-Case *Export Exec Leave*





Generalization among Actors

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

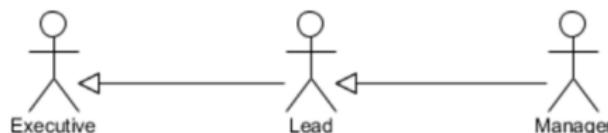
Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- There can be hierarchy among the actors, that is, nouns.
- That is the base actor will execute some use cases, and the specialized actors, will execute extra Use-Cases
- Lead is an Executive, Manager is a Lead – means that
 - Lead executes all the use cases executed by an Executive along with some additional use cases (Revoke Leave, Approve Leave)
 - Similarly, Manager executes all the Use-Cases executed by a Lead along with some additional Use-Cases (Adjust Leave, Credit Leave)



Generalization Among Actors



Use-Case Diagram: Leave Management System

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

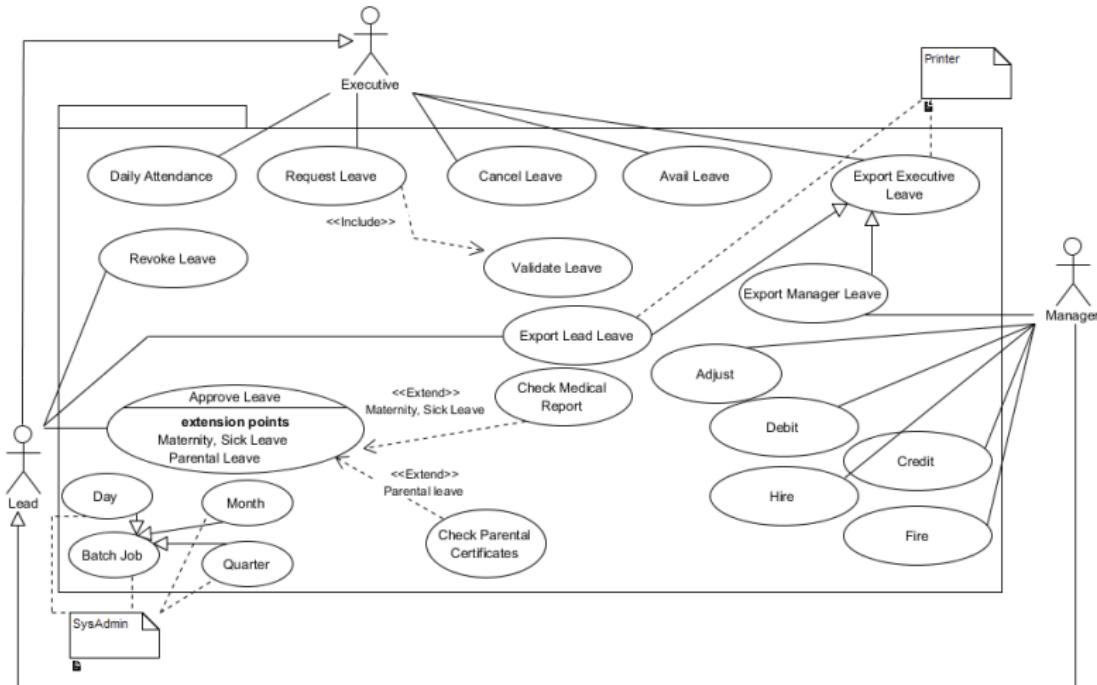
Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary



Not all use cases are shown in details



Module Summary

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

- Illustrated Use-Case diagrams for LMS



Instructor and TAs

Module 25

Partha Pratim
Das

Objectives &
Outline

Use-Cases:
RECAP

Identify Actors

Identify
Use-Cases

Identify
Relationships

Use-Case
Diagram for
LMS

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

Module 26: Object Oriented Analysis & Design

Class Diagrams: Part 1 (Class, Property & Operation)

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

● Understanding Class Diagrams



Module Outline

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

• What are Class Diagrams?

- Class
- Property (Attributes)
- Operation (Methods)
- Examples



What is a Class? RECAP (Module 13)

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

Whereas an object is a concrete entity that exists in time and space, a class represents only an abstraction, the "essence" of an object, as it were

- For a class **Faculty**, objects may be:
 - {Partha Pratim Das, Professor, CSE}
 - {Prabir Kumar Biswas, Professor, ECE}
 - {Shyamal Das Mondal, Assistant Professor, CET}
- Class **Faculty** abstracts – *Name, Designation, and Department*

A class is a set of objects share a common structure, common behavior, and common semantics

A single object is simply an instance of a class



The Canonical Form of a Complex System: RECAP (Module 04)

Module 26

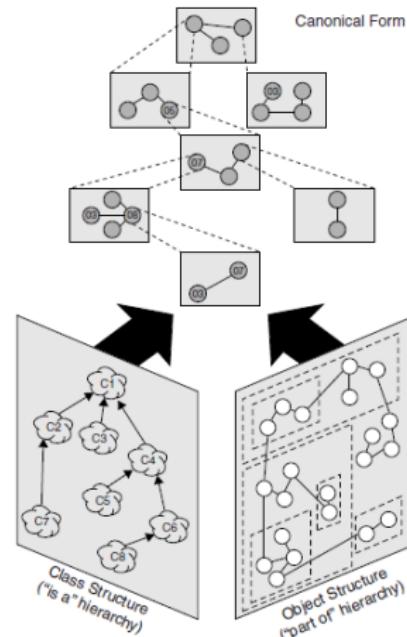
Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary



The Canonical Form of a Complex System

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Class Diagrams in SDLC phases: RECAP (Module 22)

Module 26

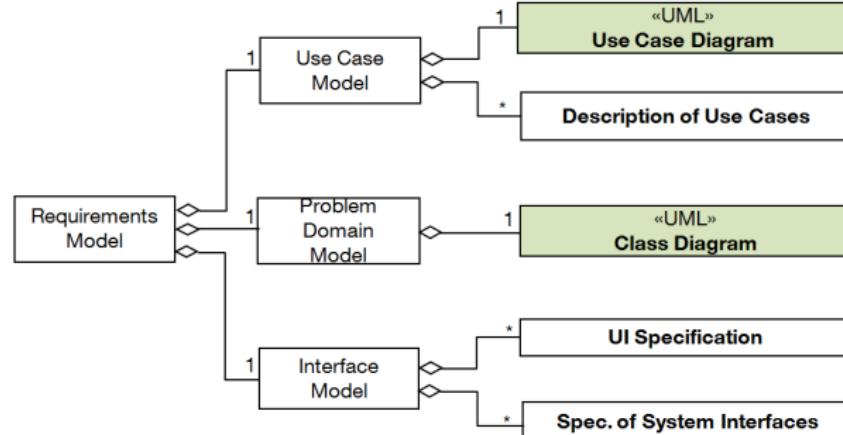
Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary



- In the **Requirements Phase**, the class diagram is used to identify the major abstractions
- At this stage the attributes and operation of each abstraction may not be known
- Classes are identified as **domain models**



Class Diagrams in SDLC phases: RECAP (Module 22)

Module 26

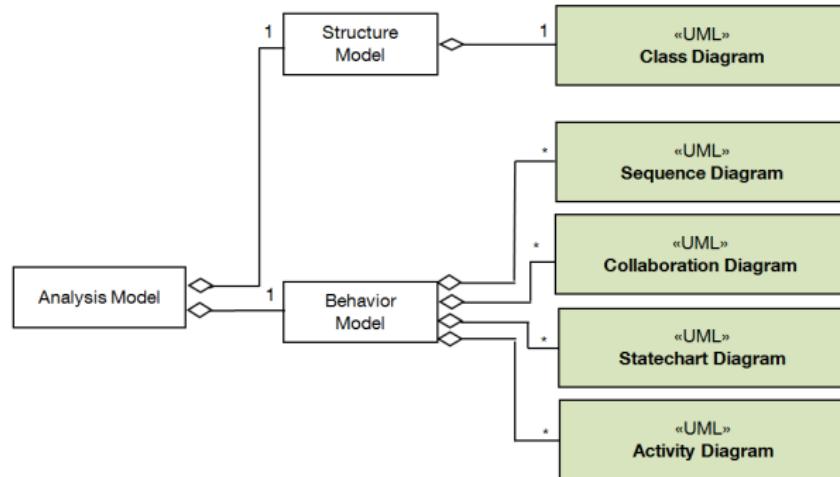
Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary



- After analysis of each abstraction, attributes and operation of each abstraction is known
- Hence the class diagram in the **Analysis Phase** is more detailed
- Classes are refined as **domain models**



Class Diagrams in SDLC phases: RECAP (Module 22)

Module 26

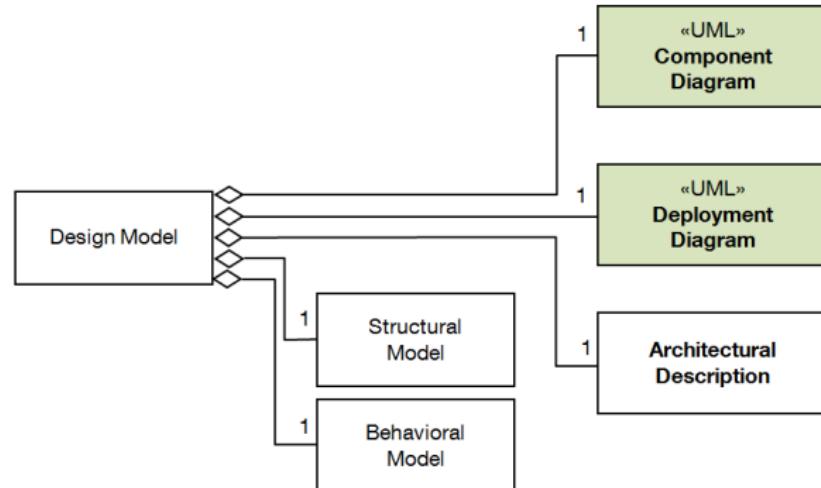
Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary



- Class diagram is included in the Structural Model
- In the **Design Phase** is further detailed
- As we engage in HLD to LLD, **implementation classes** are added



Class Diagram

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

- Class diagram is UML structure diagram which shows structure of the designed system at the level of classes and interfaces, shows their features, constraints and relationships – associations, generalizations, dependencies, etc.
- Some common types of class diagrams are:
 - Domain model diagram
 - Diagram of implementation classes

Source: *UML 2.5 Diagrams Overview:* <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Features of a class

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

- **Non Static Features:** characterizes individual instances of class
- **Static Features:** represents some characteristic of the class itself
- **Structural Features (attributes):** is a typed feature of a class that specifies the structure of instances of the class
- **Behavioral Features (Methods):** is a feature of a class that specifies an aspect of the behavior of its instances

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Notation for Class

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

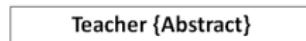
Summary

- Class name should be centered and in bold face inside a solid-outline rectangle, with the first letter of class name capitalized



Class Student - details suppressed

- Abstract Classes (which cannot be instantiated) have the keyword abstract mentioned within { }



Abstract Class Teacher - details suppressed

- A class has optional compartments separated by horizontal lines containing attributes and methods in order



Notation for Property (Attributes)

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

● **Property (Attributes) specification format:**

*Visibility PropertyName : Type [Multiplicity] = DefaultValue
{Property string}*

- The visibility of the properties are denoted by +(public), #(protected) and -(private)
- PropertyName is underlined if the Property is static
- A property may be *Read Only*, *Static*, *Ordered*, *Unique* or *Optional* (to indicate allowable null value)
- Property could have multiplicity. The multiplicity bounds constrain the size of the collection of property values. By default the maximum bound is 1
- The default-value option is an expression for the default value or values of the property
- A derived Property, designated by a preceding /, is one that can be computed from other properties, but doesn't actually exist

Student
+ name: String
+date_of_birth: Date
+roll_no: String {unique}
+/age: Integer
+subject: Subject[1..*]



Notation for Operations (Methods)

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

• Operation (Methods) specification format:

*Visibility OperationName (ParameterName : Type) :
ReturnType {Property string}*

- The visibility of the operations are denoted by +(public), #(protected) and -(private)
- OperationName is underlined if it is Static, and is italic if it is Abstract
- Return type is optional
- An operation may be *Read Only*, *Static*, *Ordered*, *Unique*, *Abstract*, *Sequential*, *Guarded* or *Concurrent*

Student
+name: String +date_of_birth: Date +roll_no: String unique +/age: Integer +subject: Subject[1..*]
#recordAttendance(): bool +getCertificates(): Certificates[*] {unique, ordered} -changeSubject(Subject s): bool +calculateAge(): Integer +bookMusicClassSlots (): bool {concurrent}



Abstract Classes of LMS

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

- We represent below the two abstract classes of LMS

Employee {Abstract}	Leave {Abstract}
<pre>+name: String +eid: String +gender: {Male, Female} +onDuty: Bool +salary: Double +doj: Date +reportsTo: String +recordAttendance():Bool +requestLeave(): Void +cancelLeave(): Void +availLeave(): Void +exportLeave(): Leave</pre>	<pre>+startDate: Date +endDate: Date +status: {New, Approved} +/isValid: Bool +type: {} +approveCond: Bool +eid: String +type(): String +approveLeave(Employee e): Bool +isValid(): Bool</pre>



Library Domain Model

Module 26

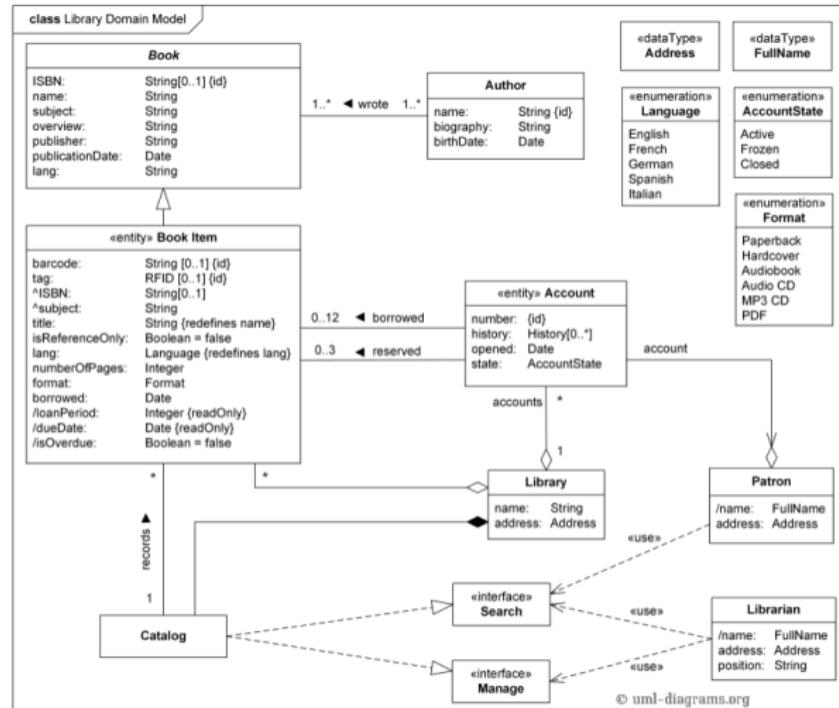
Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary



Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Library Domain Model: Annotated

Module 26

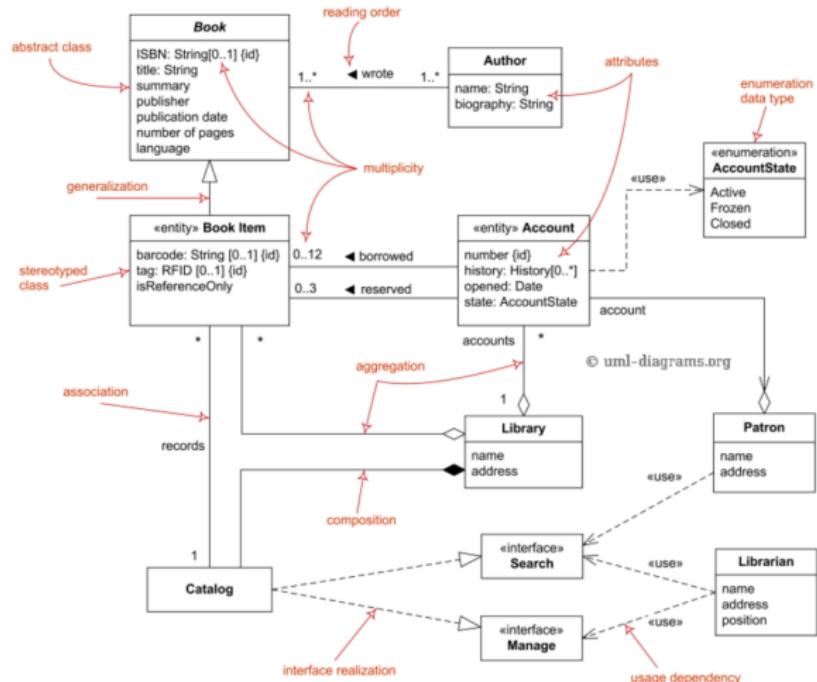
Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Module Summary

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

- Class diagrams are introduced
- Representations for properties and operations are discussed
- An example is used for detailed illustration



Instructor and TAs

Module 26

Partha Pratim
Das

Objectives &
Outline

Class
Diagrams

Class
Property
Operations
Examples

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

Module 27: Object Oriented Analysis & Design

Class Diagrams: Part 2 (Association, Weak & Strong Aggregation)

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Understanding relationships in Class Diagrams



Module Outline

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships
Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Relationships among classes
 - Association
 - Weak Aggregation
 - Strong Aggregation
 - Generalization
 - Dependency
 - Constraints



Relationships of Classes: RECAP (Module 14)

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships
Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

Class

- A daisy is a kind of flower

- A rose is a (different) kind of flower

- Red roses and yellow roses are both kinds of roses

- A petal is a part of both kinds of flowers

- Ladybugs eat certain pests such as aphids, which may be infesting certain kinds of flowers

Relationship

Sharing connection – daisies and roses are both kinds of flowers – bright colored petals, fragrance, etc.

Daisy IS_A Flower

Sharing connection – daisies and roses are both kinds of flowers ...

Rose IS_A Flower

Semantic connection – red roses and yellow roses are more alike than are daisies & roses

Red Rose IS_A Rose, Yellow Rose IS_A Rose

Semantic connection – daisies and roses are more closely related than are petals & flowers

Flower HAS_A Petal

Symbiotic connection – Ladybugs protect flowers from certain pests

Semantic Dependency

Are Roses and Candles related? – Both decorate dinner tables

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)



Association: RECAP (Module 14)

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

• Semantic Dependencies

- Most general and most semantically weak
- Bidirectional by default
- Often refined over the analysis process



Early relationship



Refined to IS_A



Early relationship



Refined to HAS_A



Early relationship

Refined to ?



Association: Notation

Module 27

Partha Pratim
Das

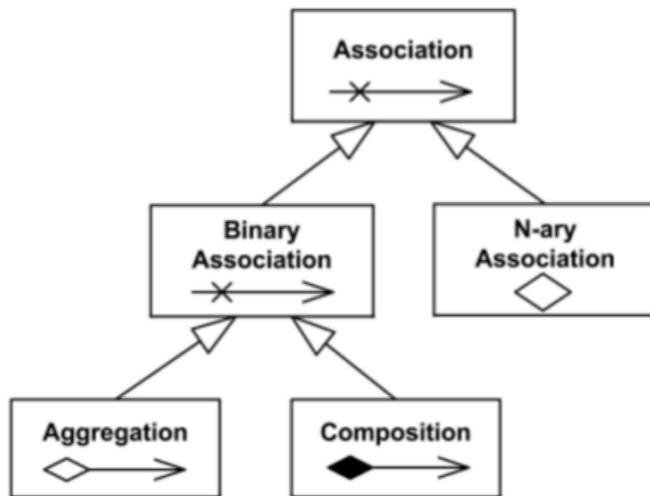
Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- An association icon (a line connector with label – association name) connects multiple classes and denotes a logical connection
- Associations can be binary or N-ary
- A class may have association to itself (Reflexive)





Association: Notation

Module 27

Partha Pratim
Das

Objectives &
Outline

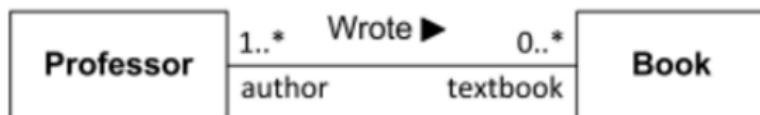
Relationships

Association

Weak
Aggregation
Strong
Aggregation
Examples

Summary

We show an association below between a Professor and a Book



An association has three main concepts

- Association End
- Navigability
- Association Arity

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Association End

Module 27

Partha Pratim
Das

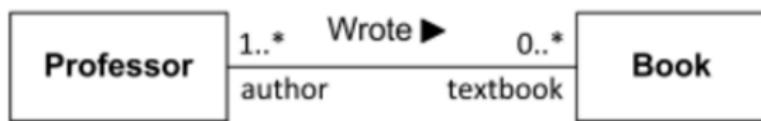
Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Association end is a connection between the line depicting an association and the icon depicting the connected classifier
- The association end name is commonly referred to as role name
- The role name is optional and suppressible



Professor "playing the role" of author is associated with textbook end typed as Book.

- Professor can have multiple roles, like author of some Books or an editor.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Association End

Module 27

Partha Pratim
Das

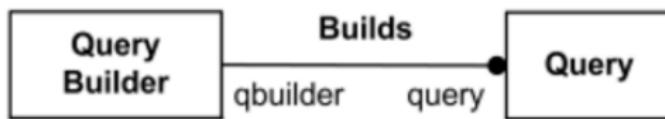
Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Association end could be owned either by **end class** or **association itself**
- Ownership of association ends by an associated classifier may be indicated graphically by a small filled circle (aka dot)



Association end query is owned by classifier QueryBuilder and
association end qbuilder is owned by association Builds itself

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Navigability

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association

Weak

Aggregation

Strong

Aggregation

Examples

Summary

- End property of association is navigable from the opposite end(s) of association if instances of the classes at this end of the link can be accessed efficiently at run-time from instances at the other ends of the link
- Navigable end is indicated by an open arrowhead on the end of an association
- Not navigable end is indicated with a small x on the end of an association

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Navigability

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary



Both ends of association have unspecified navigability.



A4 is not navigable from B4 while B4 is navigable from A4.



A2 has unspecified navigability while B2 is navigable from A2.



A5 is navigable from B5 and B5 is navigable from A5.



A3 is not navigable from B3 while B3 has unspecified navigability.



A6 is not navigable from B6 and B6 is not navigable from A6.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Arity – Binary Association

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

Each association has specific arity as it could relate two or more classes

- **Binary association** relates two typed instances
- It is normally rendered as a solid line connecting two classifiers, or a solid line connecting a single classifier to itself (the two ends are distinct)
- The line may consist of one or more connected segments



Job and Year classes are associated

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Arity – Binary Association

Module 27

Partha Pratim
Das

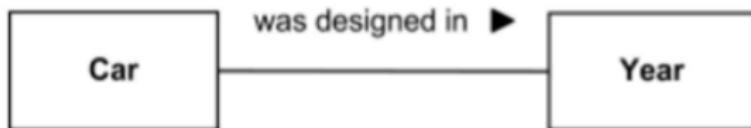
Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- A small solid triangle could be placed next to or in place of the name of binary association (drawn as a solid line) to show the order of the ends of the association
- The arrow points along the line in the direction of the last end in the order of the association ends



Order of the ends and reading: Car - was designed in - Year

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Arity – N-ary Association

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association

Weak

Aggregation

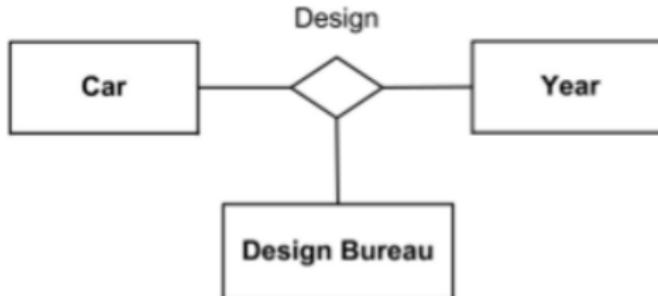
Strong

Aggregation

Examples

Summary

- **N-ary association** may be drawn as a diamond (larger than a terminator on a line) with a solid line for each association end connecting the diamond to the classifier that is the end's type
- N-ary association with more than two ends can only be drawn the following way



Ternary association Design relates three classes

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Health-care Organization Model

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

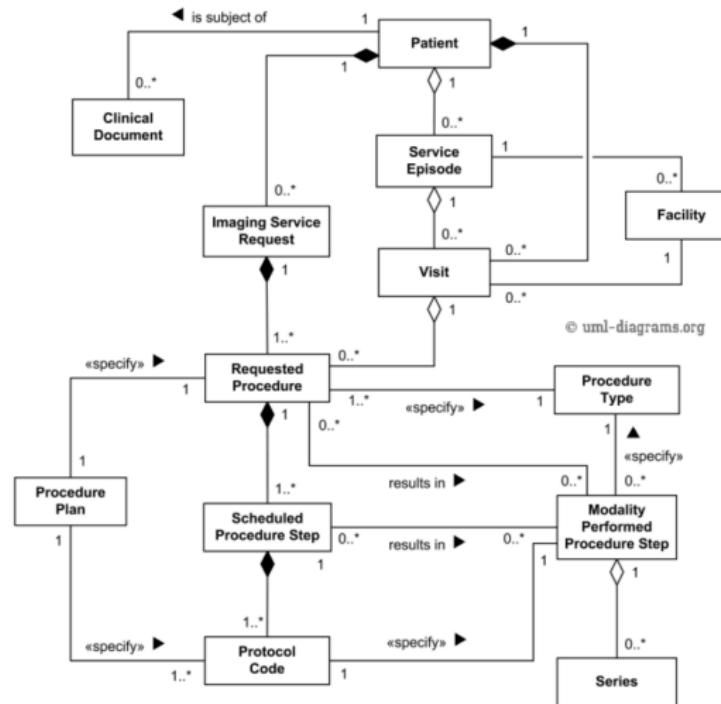
Association

Weak
Aggregation

Strong
Aggregation

Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Associations in LMS

Module 27

Partha Pratim
Das

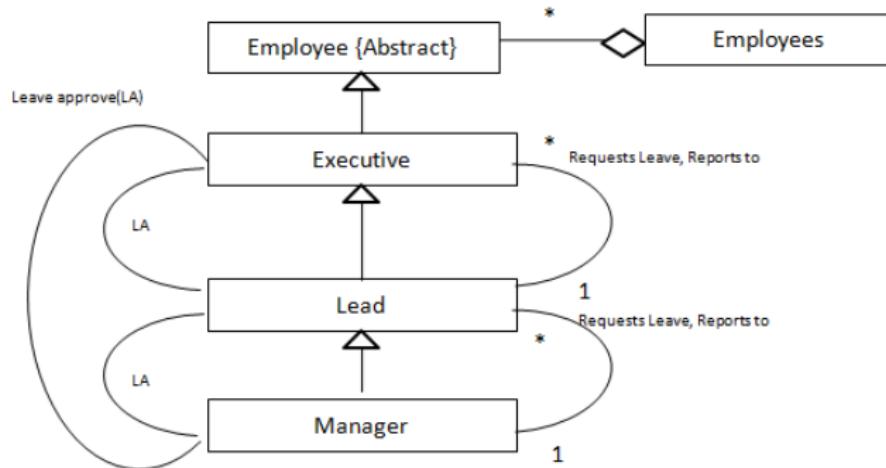
Objectives &
Outline

Relationships

Association

Weak
Aggregation
Strong
Aggregation
Examples

Summary



Associations in LMS



Aggregation (HAS_A): RECAP (Module 14)

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association

Weak

Aggregation

Strong

Aggregation

Examples

Summary

- Whole / Part relationships

- Say, we model **Flower HAS_A Petal**
- Flower contains many Petals**
- Flower is the Whole, Petal is the Part**
- Depicted as:



- Physical Containment – Composition / Strong Aggregation**

- Member relationship

- Say, we model **Library HAS Users**
- Library enrolls many Users**
- Library does not contain the Users**
- Depicted as:



- Conceptual Containment – Weak Aggregation**



Weak Aggregation

Module 27

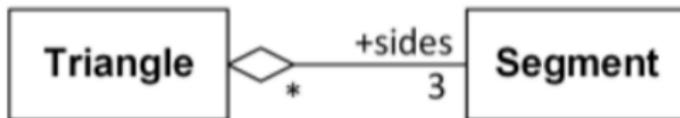
Partha Pratim
Das

Objectives &
Outline

Relationships
Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Weak aggregation is depicted as an association decorated with a hollow diamond at the aggregate end of the association line



Triangle has 'sides' collection of three line Segments

Each line Segment could be part of none, one, or several triangles

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Weak Aggregation

Module 27

Partha Pratim
Das

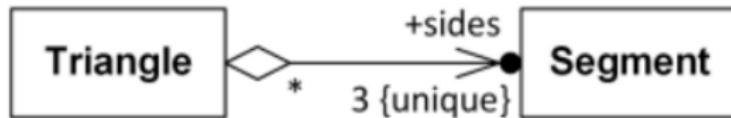
Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Weak aggregation could be depicted together with navigability and association end ownership



Triangle has 'sides' collection of three unique line Segments.
Line segments are navigable from Triangle.

Association end 'sides' is owned by Triangle, not by association itself

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Strong Aggregation (Composition)

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association

Weak

Aggregation

Strong

Aggregation

Examples

Summary

- Strong aggregation (Composition) is depicted as a binary association decorated with a filled black diamond at the aggregate (whole) end



Folder could contain many files, while each File has exactly one Folder parent

If Folder is deleted, all contained Files are deleted as well

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Library Domain Model

Module 27

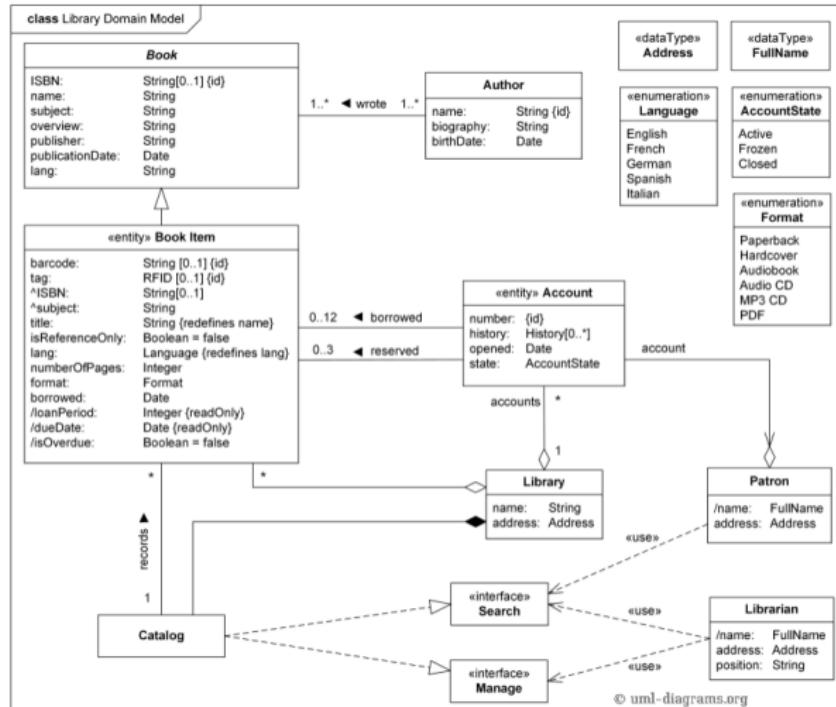
Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Library Domain Model: Annotated

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association

Weak

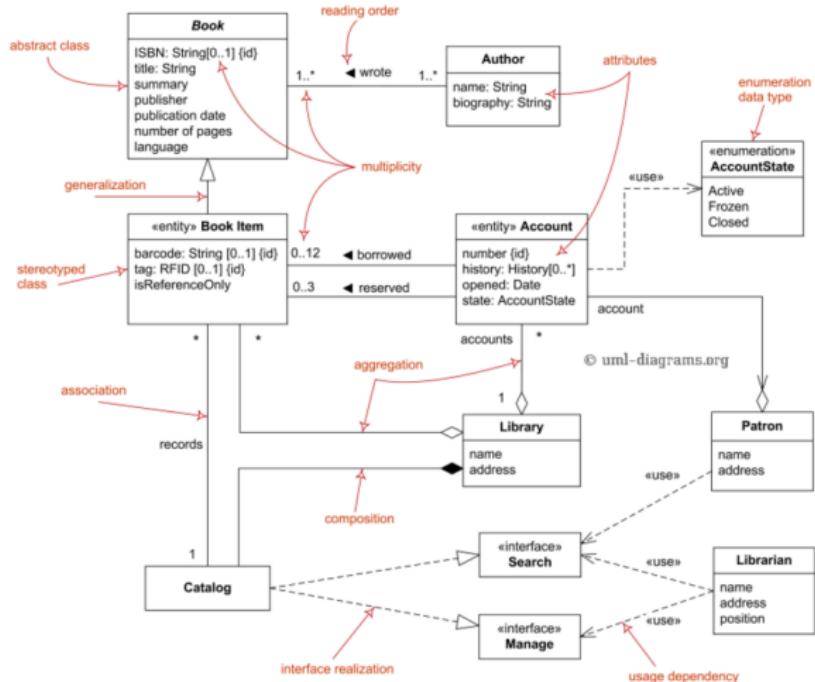
Aggregation

Strong

Aggregation

Examples

Summary



Domain diagram overview - classes, interfaces, associations, usage, realization, multiplicity.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Module Summary

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships

Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

- Association Relationships among classes are discussed
- Weak Aggregation and Strong Aggregation are important binary associations



Instructor and TAs

Module 27

Partha Pratim
Das

Objectives &
Outline

Relationships
Association
Weak
Aggregation
Strong
Aggregation
Examples

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

Module 28: Object Oriented Analysis & Design

Class Diagrams: Part 3

(Generalization, Dependency & Constraints)

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in
Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

- Understanding relationships in Class Diagrams



Module Outline

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

- Relationships among classes
 - Association
 - Weak Aggregation
 - Strong Aggregation
 - Generalization
 - Dependency
 - Constraints
- Class Diagram for LMS



Inheritance (IS_A): RECAP (Module 14)

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization

Dependency

Constraints

Examples

LMS Class
Diagram

Summary

- Generalization / Specialization relationships
 - Say, we model **Daisy IS_A Flower**
 - **Daisy** will **inherit** the properties of **Flower**, and have some more of its own
 - **Flower** is the **Generalization**
 - **Daisy** is the **Specialization**
 - Depicted as:



- Semantically most interesting
- Can *delegate* behavior to related objects
- Comes in a number of flavors
 - Single / Multilevel / Hierarchical Inheritance
 - Multiple Inheritance
 - Hybrid Inheritance



Generalization

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization

Dependency

Constraints

Examples

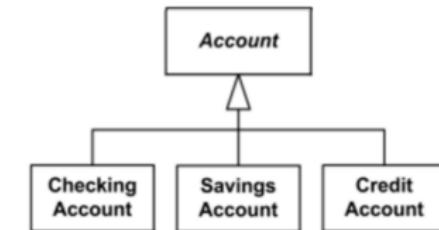
LMS Class
Diagram

Summary

- A generalization is shown as a line with a hollow triangle as an arrowhead



Separate target style



Shared target style

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Multiple Inheritance: RECAP (Module 14)

Module 28

Partha Pratim
Das

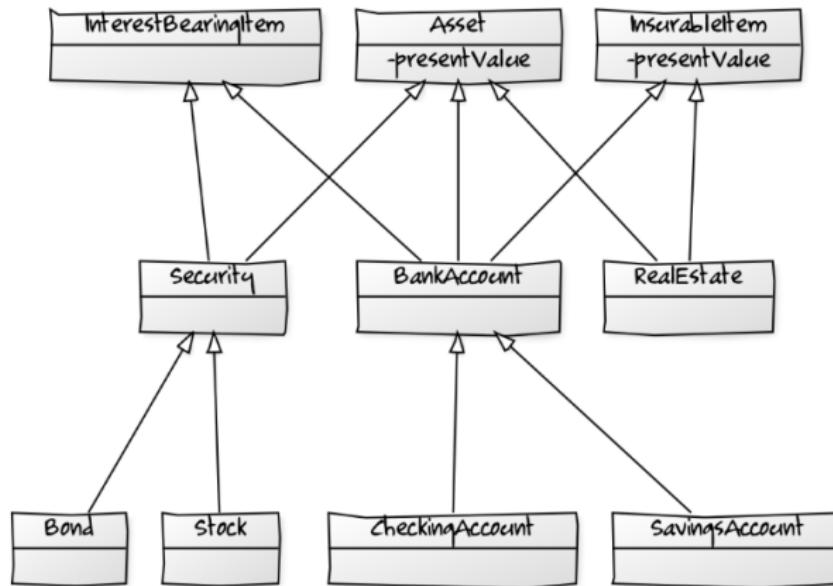
Objectives &
Outline

Relationships

Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary



- More than one **superclass** for a **subclass**
- RealEstate IS_A Asset, InsurableItem**



Multiple Inheritance

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization

Dependency

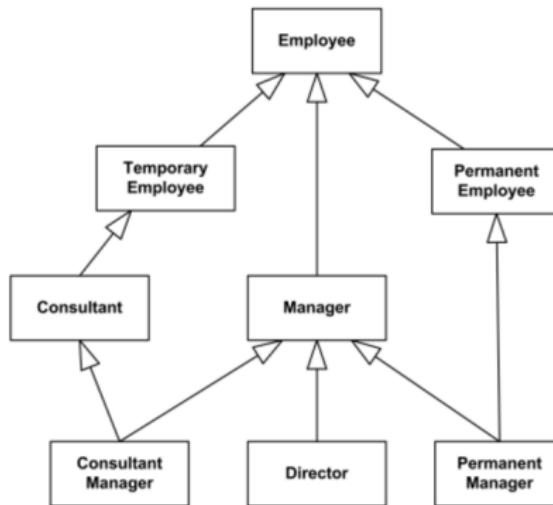
Constraints

Examples

LMS Class
Diagram

Summary

- Multiple inheritance is implicitly allowed by UML standard, while the standard provides no definition of what it is



Multiple inheritance for Consultant Manager and Permanent Manager – both inherit from two classes

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Dependency

Module 28

Partha Pratim
Das

Objectives &
Outline

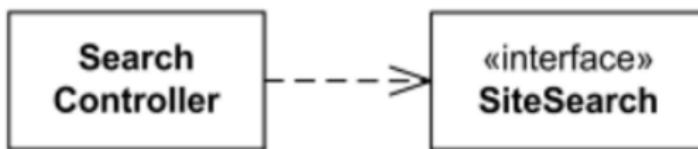
Relationships

Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

- Dependency is a directed relationship which is used to show that some UML element or a set of elements requires, needs or depends on other model elements for specification or implementation



Class SearchController depends on (requires) SiteSearch interface

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Constraints

Module 28

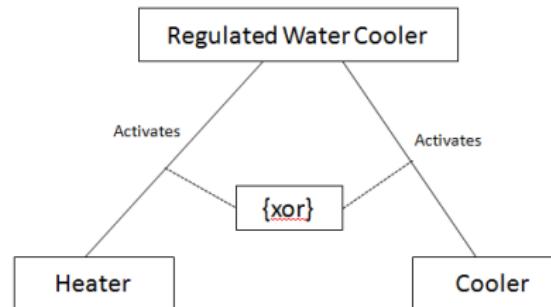
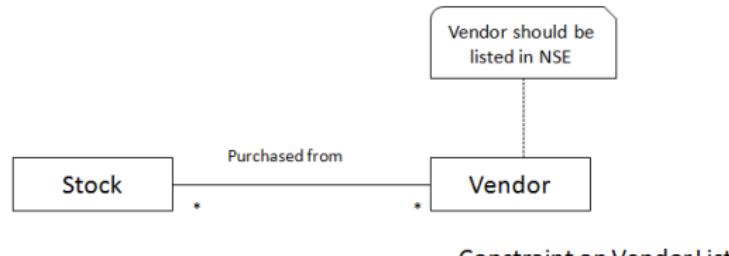
Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary



Constraint on Activation of Heater and Cooler



Library Domain Model

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization

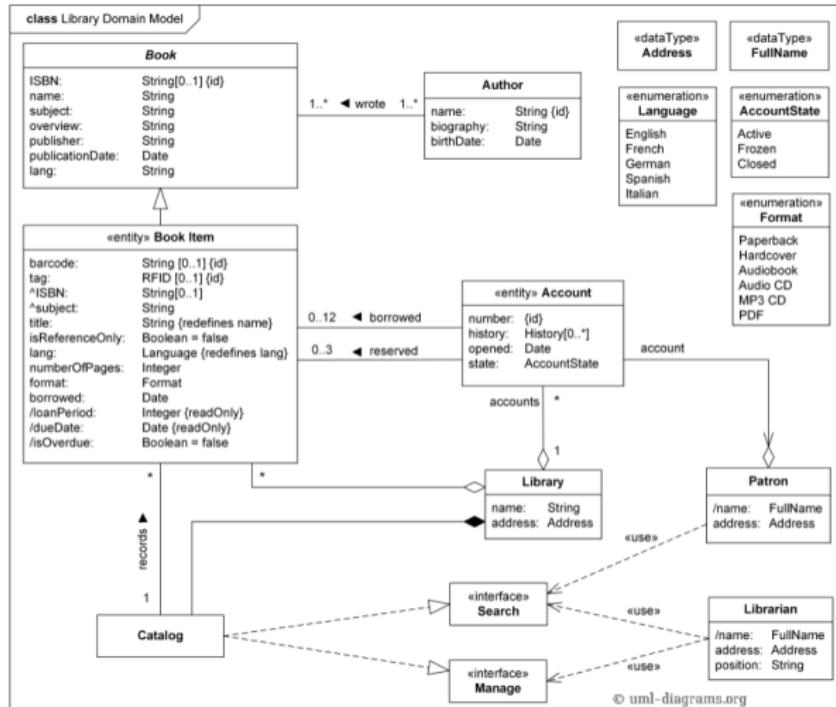
Dependency

Constraints

Examples

LMS Class
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Library Domain Model: Annotated

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization

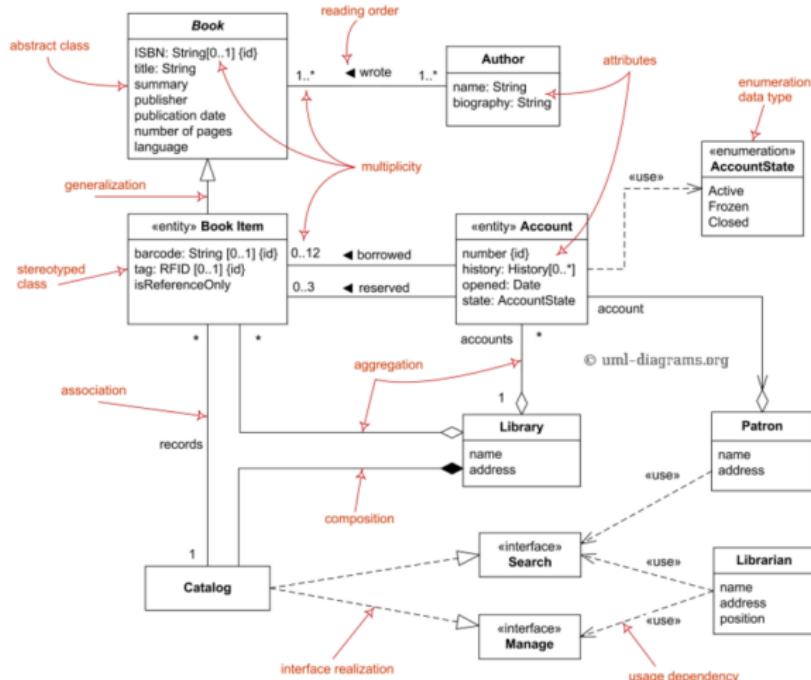
Dependency

Constraints

Examples

LMS Class
Diagram

Summary



Domain diagram overview - classes, interfaces, associations, usage, realization, multiplicity.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (17-Aug-16)



Use-Case Diagram for LMS RECAP (Module 25)

Module 28

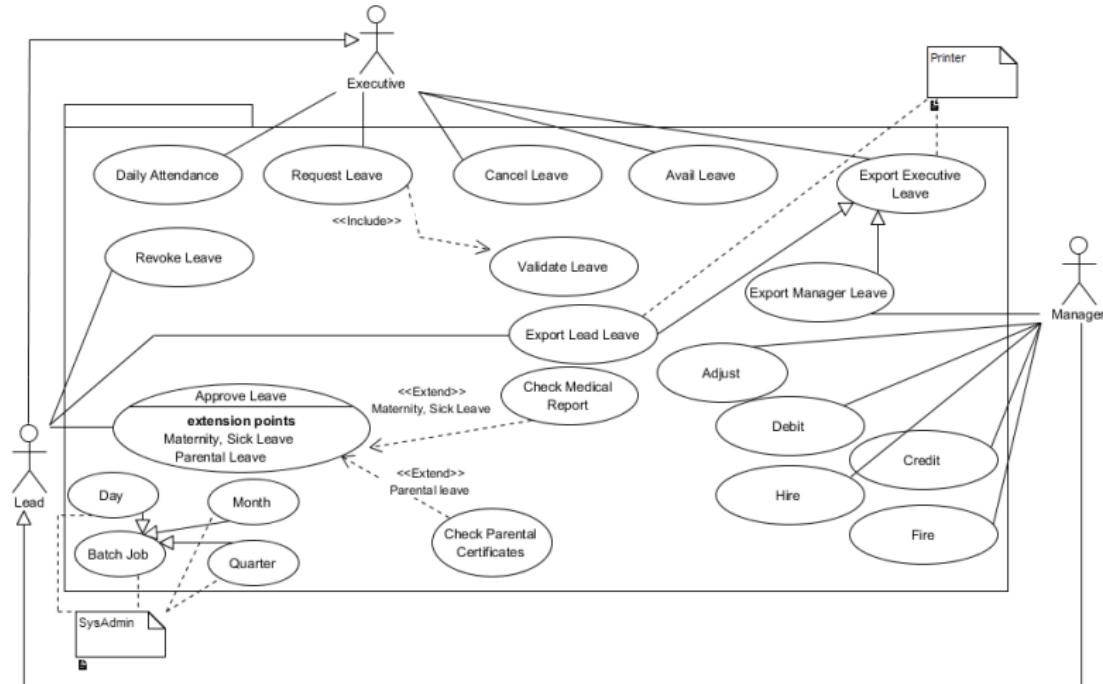
Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary



Not all use cases are shown in details



Class Diagram for LMS

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

We now derive the Class Diagram for LMS. The steps involved are:

- Identify Classes {Abstract Classes}
- Identify Properties and Operations
- Identify the Relationships among Classes
- Class Diagram



Identification of Classes {Abstract Classes}

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

- Reading through the specification of the Leave Management System, we identify the various instances, that is, objects
- We categorize them into two abstract classes: Employee and Leave

Employee {Abstract}

Leave {Abstract}



Identification of Properties

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

Properties of the two abstract class of LMS

Employee {Abstract}	Leave {Abstract}
+name: String +eid: String +gender: {Male, Female} +onDuty: Bool +salary: Double +doj: Date +reportsTo: String	+startDate: Date +endDate: Date +status: {New, Approved} +/isValid: Bool +type: {} +approveCond: Bool +eid: String



Identification of Operations

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

Employee {Abstract}	Leave {Abstract}
<pre>+name: String +eid: String +gender: {Male, Female} +onDuty: Bool +salary: Double +doj: Date +reportsTo: String +recordAttendance():Bool +requestLeave(): Void +cancelLeave(): Void +availLeave(): Void +exportLeave(): Leave</pre>	<pre>+startDate: Date +endDate: Date +status: {New, Approved} +/isValid: Bool +type: {} +approveCond: Bool +eid: String +type(): String +approveLeave(Employee e): Bool +isValid(): Bool</pre>



Identification of Associations

Module 28

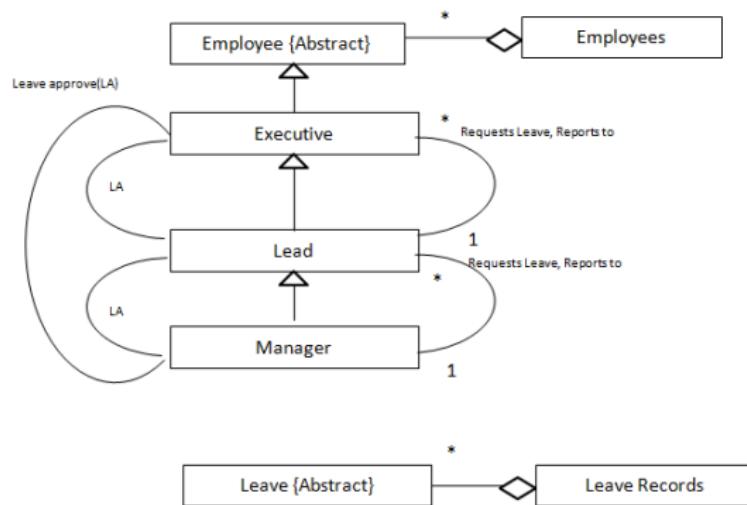
Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary





Identification of Generalizations

Module 28

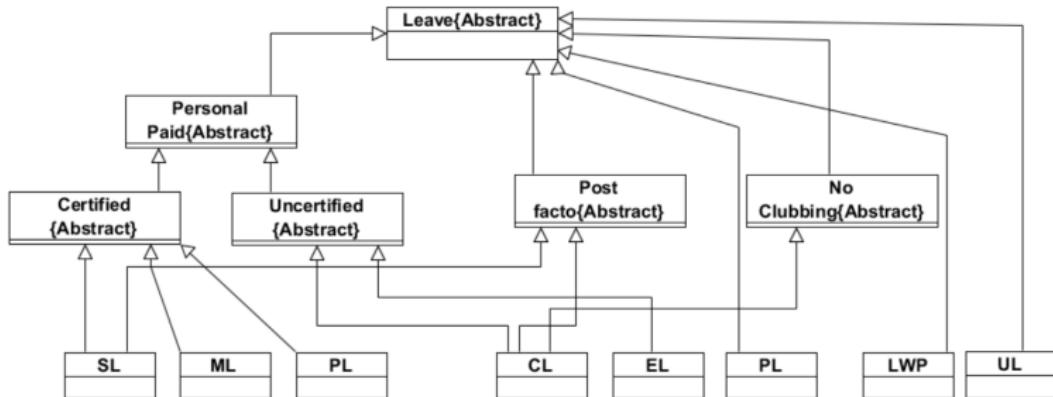
Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary





LMS Class Diagram (Partial)

Module 28

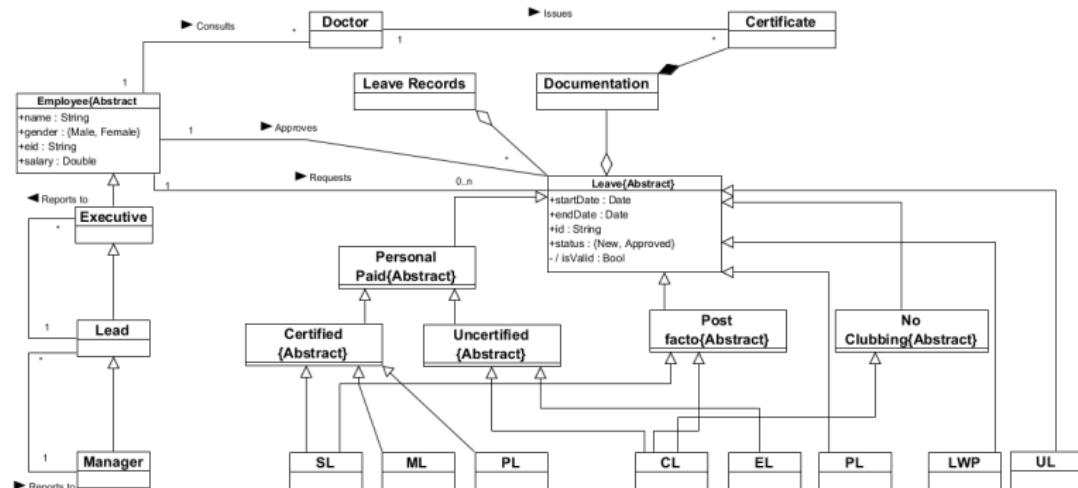
Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary





Module Summary

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships

Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

- Discussed Generalization, Dependency and Constraint relationships
- A partial Class Diagram for the Leave Management System (LMS)



Instructor and TAs

Module 28

Partha Pratim
Das

Objectives &
Outline

Relationships
Generalization
Dependency
Constraints
Examples

LMS Class
Diagram

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

Module 29: Object Oriented Analysis & Design

Sequence Diagrams: Part 1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

● Understanding Sequence Diagrams



Module Outline

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

● What are Sequence Diagrams?

- Lifeline
- Messages
- Interaction Fragments
- Examples



Client-Server Computing Model: RECAP (Modules 05, 11)

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

- No object exists in isolation
- Objects are acted on and themselves act on other objects
- Leads to the **Client-Server Model** of computing where
 - Behavior is
 - Services provided by an object
 - Services are requested by
 - Sending Messages, Invoking Operations
 - In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



Sequence Diagrams in SDLC phases: RECAP (Module 22)

Module 29

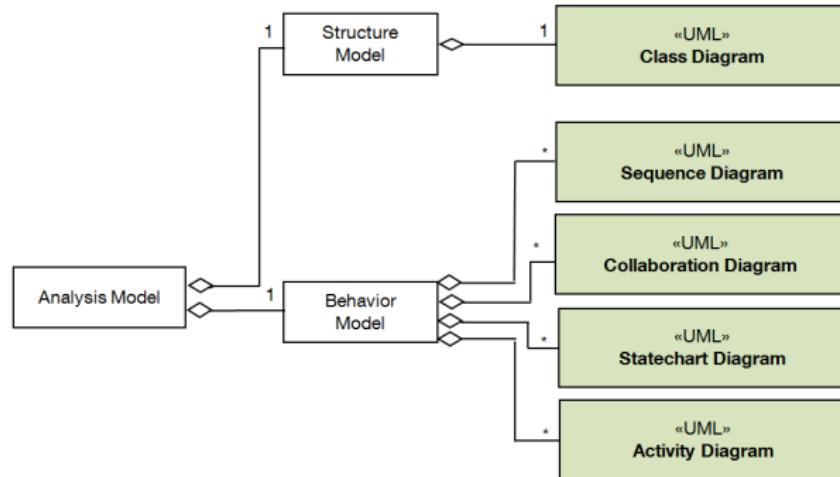
Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



- In the **Analysis Phase** the problem domain is analyzed and refined from the **Requirements Phase**
- The behavior model of the system is hence understood in this phase
- Sequence diagram is a major result of the Analysis Phase



Sequence Diagrams in SDLC phases: RECAP (Module 22)

Module 29

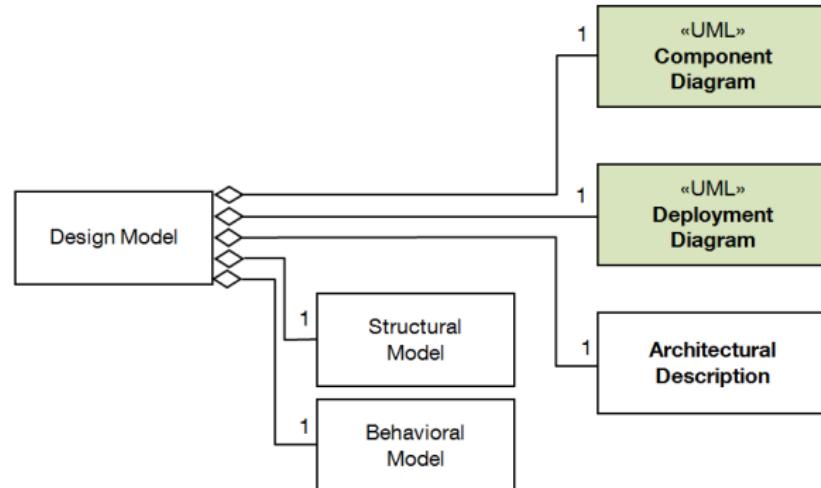
Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



- Sequence diagram is included in the Behavioral Model
- It is further refined in the **Design Phase**



What are Sequence Diagrams?

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

- Sequence diagram is the most common kind of Interaction diagram, which focuses on the message interchange between a number of lifelines
- Sequence diagram is a UML behavior diagram
- Sequence diagram depicts the inter-object behavior of a system, ordered by time
- The major components of a Sequence Diagram are:
 - Lifeline
 - Messages
 - Interaction Fragments



Lifeline

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

- **Lifeline** is an element which represents an individual participant in the interaction
- Lifelines represent only one interacting entity
- If the referenced connectable element is multi-valued (that is, has a multiplicity > 1), then the lifeline may have an expression (selector) that specifies which particular part is represented by this lifeline
- A lifeline is shown using a symbol that consists of a rectangle forming its "head" followed by a vertical line (which may be dashed) that represents the lifetime of the participant
- The information identifying a lifeline is depicted as
ObjectName[selector]:ClassName

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Lifeline

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



Lifeline "data" of class Stock



Anonymous lifeline of class User



Lifeline "x" of class X is selected with selector [k]

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Named Elements of LMS

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

- The major named elements of LMS are Employee and Leave. Few instances of them shown below.



- The major interaction activity of LMS is **Request Leave**, **Approve Leave** which requires interaction between the two major classes, Employee and Leave



Types of Messages

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

- Message is an element that defines one specific kind of communication between lifelines of an interaction
 - There are 2 major types of message in Sequence Diagram
 - Messages by Action Type
 - Messages by Presence of Events
- Message by Action Type:** A message reflects either an operation call and start of execution or a sending and reception of a signal
- Message by Presence of Events:** A message depends on whether message send event and receive events are present

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Messages by Action Type

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

The various types of Messages by Action type are:

- synchronous call
- asynchronous call / signal
- create
- delete
- reply



Messages by Action Type

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

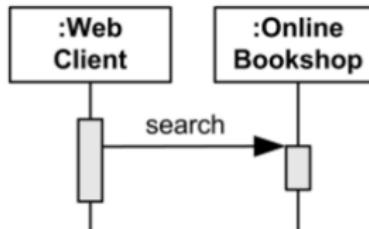
Summary

Synchronous call typically represents operation call - send message and suspend execution while waiting for response

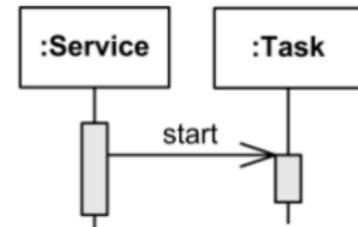
Notation: filled arrow head.

Asynchronous call - send message and proceed immediately without waiting for return value

Notation: Open arrow head



Web Client searches Online Bookshop and waits
for results



Service starts Task and proceeds in parallel
without waiting

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Messages by Action Type

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

Create message is sent to a lifeline to create itself

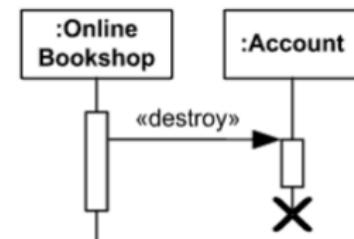
Notation: Dashed line with open arrowhead



Online Bookshop creates Account

Delete message is sent to terminate another lifeline

Notation: lifeline usually ends with a cross (X) at the bottom



Online Bookshop terminates Account

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Messages by Action Type

Module 29

Partha Pratim
Das

Objectives &
Outline

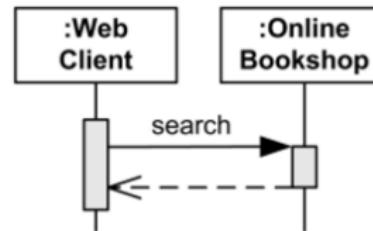
Sequence
Diagrams

Lifeline
Messages
Examples

Summary

Reply message to an operation call

Notation: Dashed line with open arrow head



Web Client searches Online Bookshop and waits for results to be returned

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Messages by Presence of Events

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

The various types of Messages by Presence of Events are:

- complete message
 - The semantics of a complete message is the trace $\langle \text{sendEvent}, \text{receiveEvent} \rangle$
 - Both sendEvent and receiveEvent are present
- lost message
- found message
- unknown message (default) – both sendEvent and receiveEvent are absent (should not appear)



Messages by Presence of Events

Module 29

Partha Pratim
Das

Objectives &
Outline

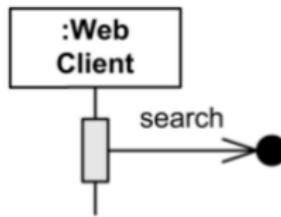
Sequence
Diagrams

Lifeline
Messages
Examples

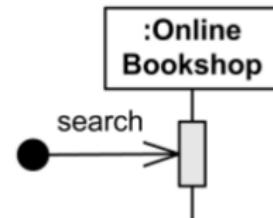
Summary

Lost Message is a message where the sending event is known, but there is no receiving event

Found Message is a message where the receiving event is known, but there is no (known) sending event



Web Client sent search message which was lost



Online Bookshop gets search message of unknown origin

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



An Annotated Sequence Diagram

Module 29

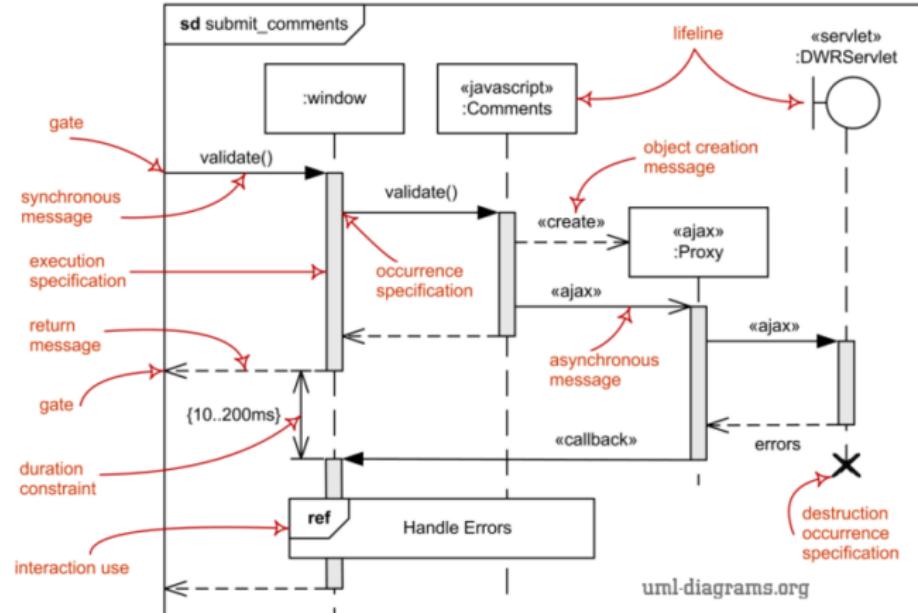
Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Example: Login

Module 29

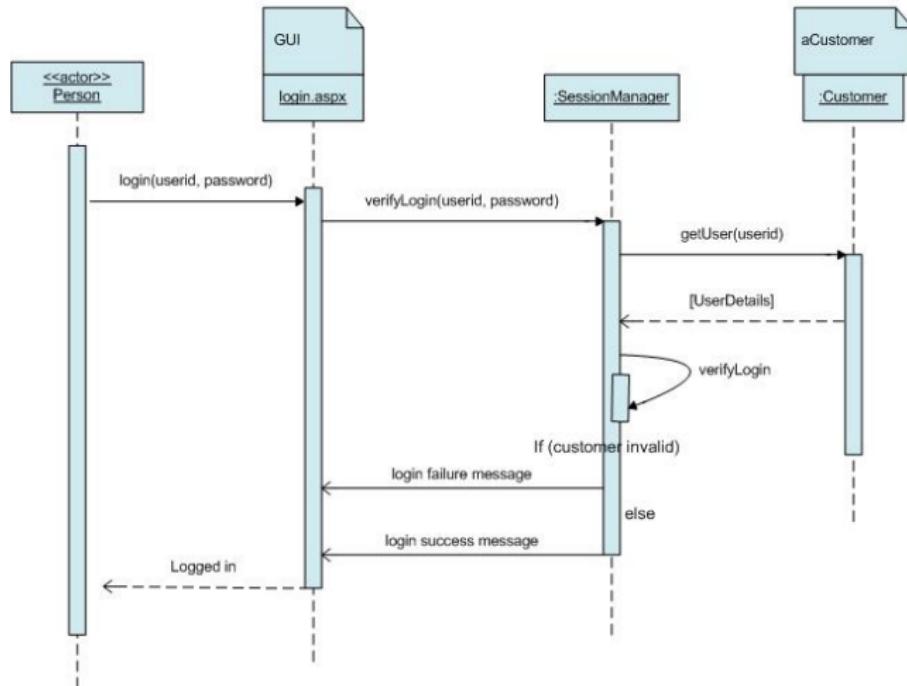
Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



Source: http://people.cs.ksu.edu/~reshma/798_SequenceDiagram.htm (18-Aug-16)



Example: Place Order

Module 29

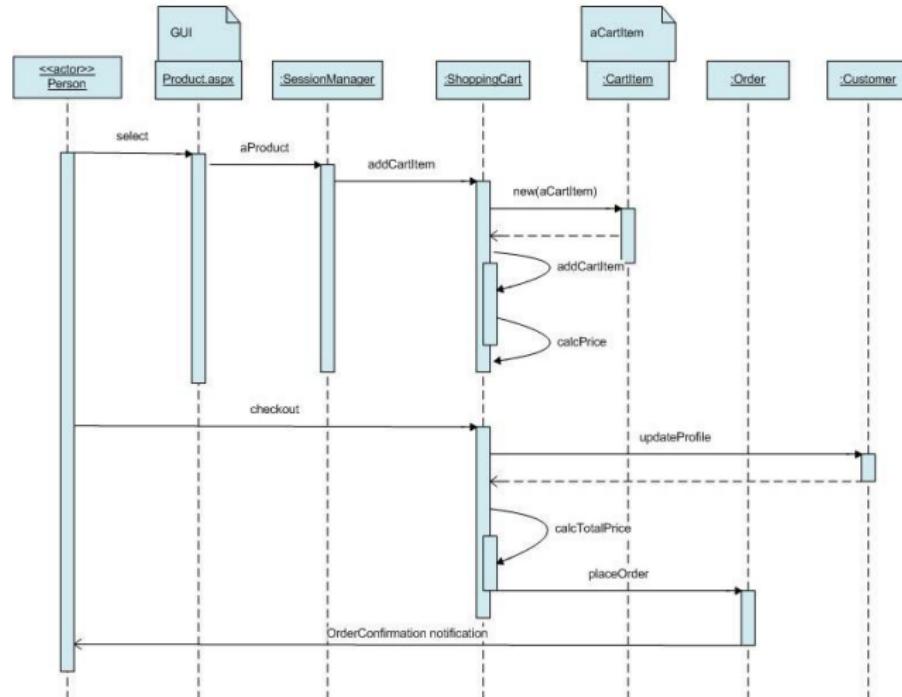
Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



Source: http://people.cs.ksu.edu/~reshma/798_SequenceDiagram.htm (18-Aug-16)



Example: Facebook Authentication

Module 29

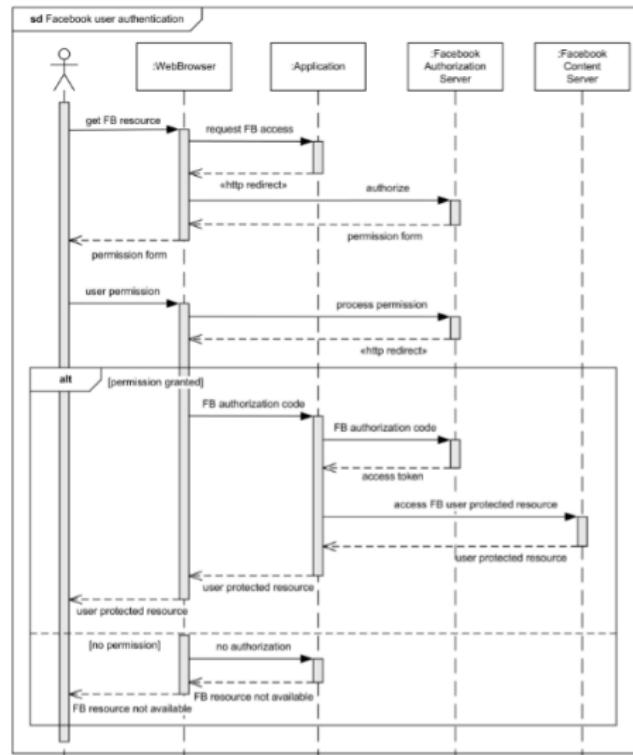
Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Messages of LMS

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

The messages for the major activities of LMS are given below:

● Request Leave

- Request Leave() from Employee
- new() Leave
- isValid() Leave
- return(ifvalid == true)

● Approve Leave

- Approve Leave() from Employee
- Approver()
- Reportingto()
- return(Reportingto)



Module Summary

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

- Introduced sequence diagram to capture the detailed execution flows of objects, their interactions and lifeline with a temporal ordering among events
- Discussed lifeline and messages in depth with examples



Instructor and TAs

Module 29

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagrams

Lifeline
Messages
Examples

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary

Module 30: Object Oriented Analysis & Design

Sequence Diagrams: Part 2

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments

Occurrence
Execution

State Invariant
Interaction Use

LMS Sequence
Diagram

Summary

- Understanding Sequence Diagrams
- Sequence diagram for Leave Management System (LMS)



Module Outline

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary

- What are Sequence Diagrams?
 - Lifeline
 - Messages
 - Interaction Fragments
 - Examples
- Sequence Diagram for LMS



Sequence Diagrams: RECAP (Module 29)

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary

- The various objects of the system interact with each other, through exchange of messages to invoke the various operations of the object.
- Sequence diagram is a major diagram to depict the inter object behaviour of a system, ordered by time.
- Sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between a number of lifelines.
- The major components of a Sequence Diagram are
 - Lifeline
 - Messages
 - Interaction Fragments



Example: Login: RECAP (Module 29)

Module 30

Partha Pratim
Das

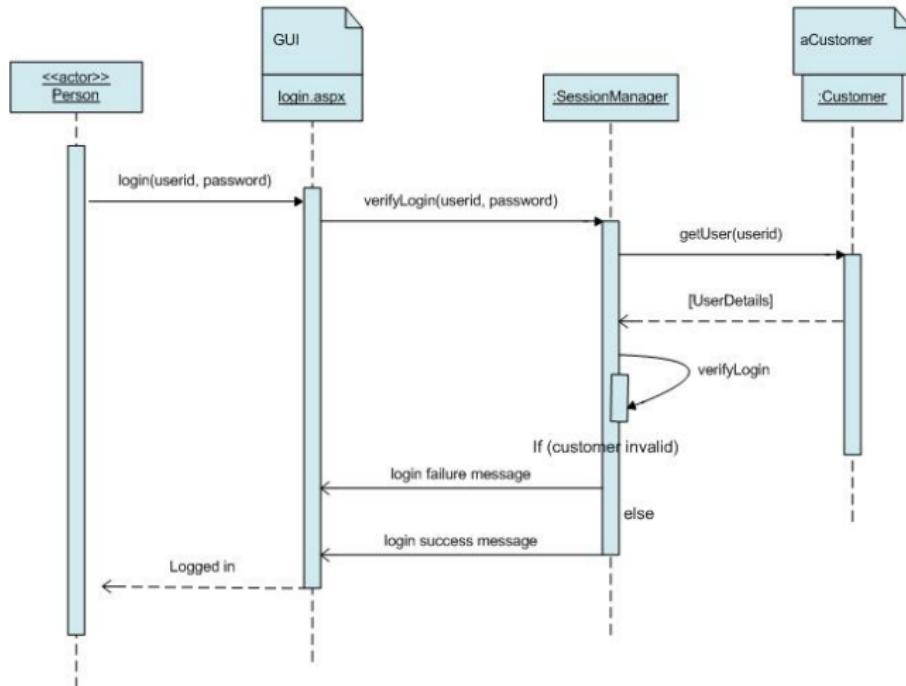
Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary



Source: http://people.cs.ksu.edu/~reshma/798_SequenceDiagram.htm (18-Aug-16)



Example: Place Order: RECAP (Module 29)

Module 30

Partha Pratim
Das

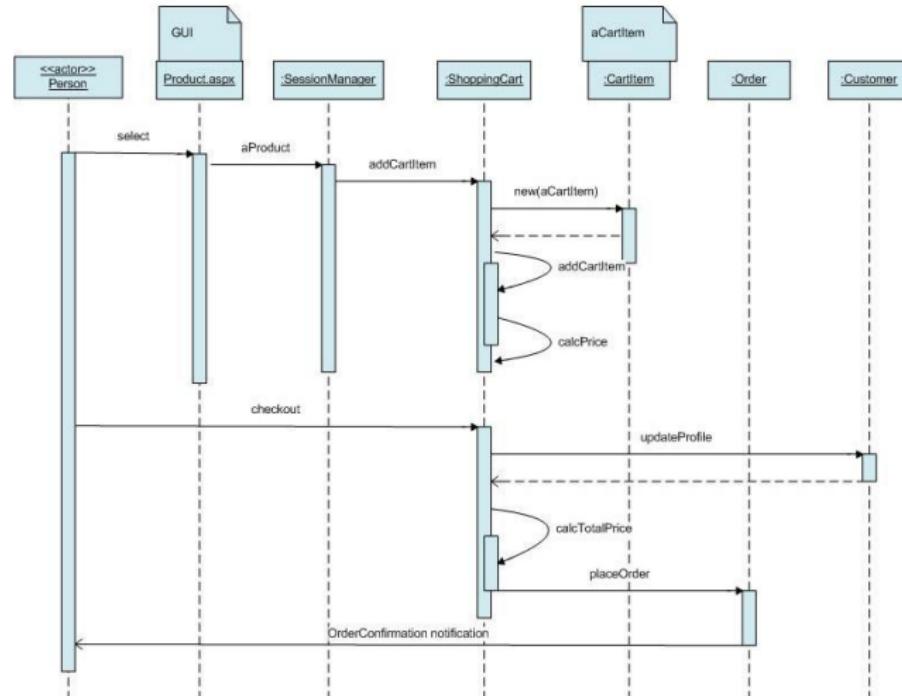
Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary



Source: http://people.cs.ksu.edu/reshma/798_SequenceDiagram.htm (18-Aug-16)



Interaction Fragments

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments

Occurrence
Execution
State invariant
Interaction Use

LMS Sequence
Diagram

Summary

- Interaction fragment is a named element representing the most general interaction unit
- Each interaction fragment is conceptually like an interaction by itself
- There is no general notation for an interaction fragment. Its sub-classes define their own notation
- Examples of Interaction Fragments include:
 - Occurrence
 - Execution
 - State invariant
 - Combined fragment
 - Interaction use

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Interaction Fragment: Occurrence

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments

Occurrence
Execution

State Invariant
Interaction Use

LMS Sequence
Diagram

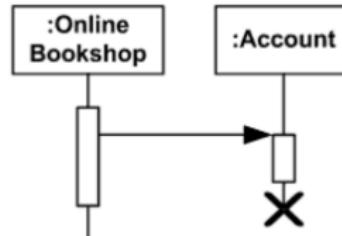
Summary

Occurrence is interaction fragment which represents a moment in time (event) at the beginning or end of a message or at the beginning or end of an execution

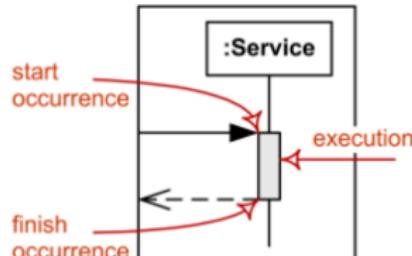
Message occurrence represents events as sending and receiving of signals

Destruction occurrence destruction of the instance described by the life-line

Execution occurrence represents moments in time at which actions or behaviors start or finish.



Account lifeline is terminated



Duration of an execution is represented by two execution occurrences - start and finish

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Interaction Fragment: Execution

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution

State Invariant
Interaction Use

LMS Sequence
Diagram

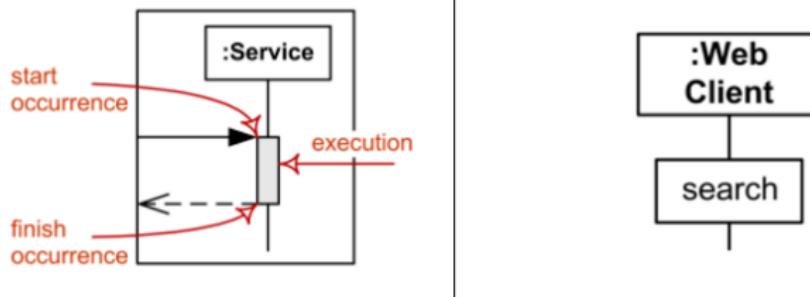
Summary

Execution (Activation) is an interaction fragment which represents a period in the participant's lifetime when it is

- executing a unit of behavior or action within the lifeline, or
- sending a signal to another participant, or
- waiting for a reply message from another participant

Execution is represented as a thin grey or white rectangle on the life-line

Execution can be represented by a wider labeled rectangle, where the label identifies the action



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Interaction Fragment: Execution

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

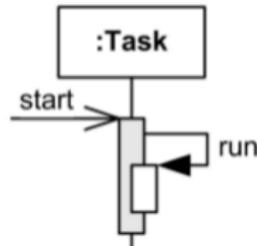
Interaction
Fragments
Occurrence
Execution

State Invariant
Interaction Use

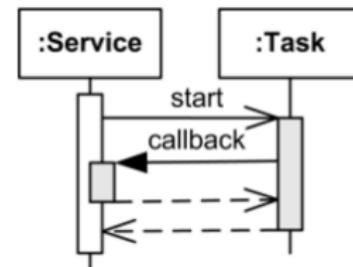
LMS Sequence
Diagram

Summary

Overlapping execution specifications on the same lifeline are represented by overlapping rectangles



Overlapping execution specifications on the same
lifeline - message to self



Overlapping execution specifications on the same
lifeline - callback message.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Interaction Fragment: State Invariant

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

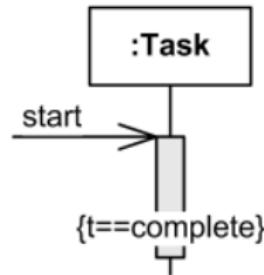
LMS Sequence
Diagram

Summary

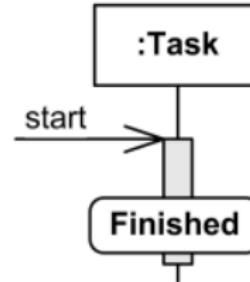
State Invariant is an interaction fragment which represents a run-time constraint on the participants of the interaction. It may be used to specify different kinds of constraints, such as values of attributes or variables, internal / external states, etc.

State invariant is usually shown as a constraint in curly braces on the lifeline

State invariant may be shown as a state symbol



Attribute t of Task should be equal to complete



Task should be in Finished state

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Interaction Fragment: Interaction Use

Module 30

Partha Pratim
Das

Objectives &
Outline

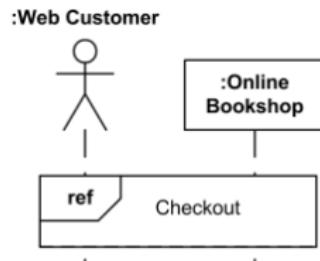
Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

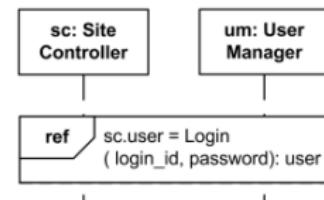
LMS Sequence
Diagram

Summary

Interaction Use is an interaction fragment which allows to use (or call) another interaction. Large and complex sequence diagrams could be simplified with interaction uses. It is also common to reuse some interaction between several other interactions



Web customer and Bookshop use (reference)
interaction Checkout



Use Login interaction to authenticate user and
assign result back to the user attribute of Site
Controller

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



An Annotated Sequence Diagram

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments

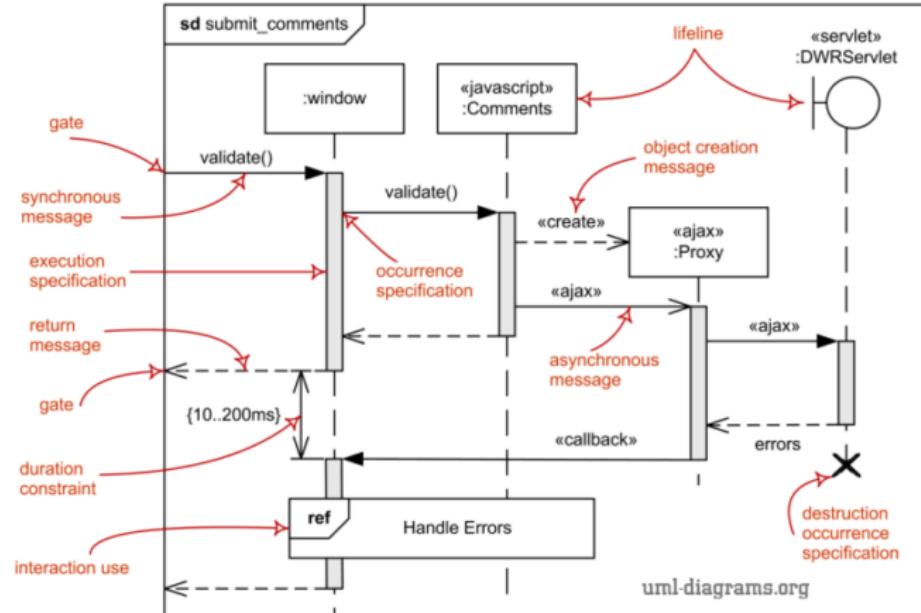
Occurrence
Execution

State Invariant

Interaction Use

LMS Sequence
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (18-Aug-16)



Example: Facebook Authentication

Module 30

Partha Pratim
Das

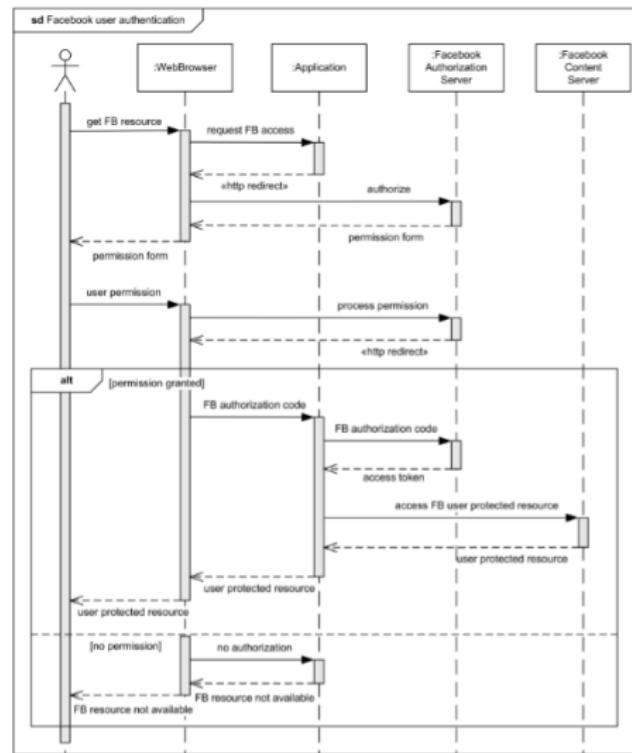
Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary





Identifying the Major Elements and their Lifelines of the Sequence Diagram

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments

Occurrence
Execution

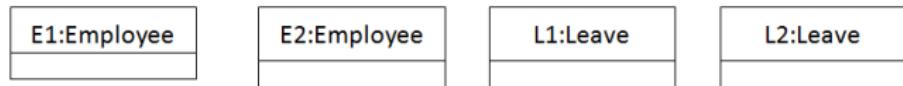
State Invariant

Interaction Use

LMS Sequence
Diagram

Summary

- Reading through the specification of the Leave Management System, we identify the major identifying elements for LMS : Employee and Leave.



- In addition, we have an LR class (Leave Record / Repository) to maintain all leave data



Sequence Diagram for LMS: Request Leave

Module 30

Partha Pratim
Das

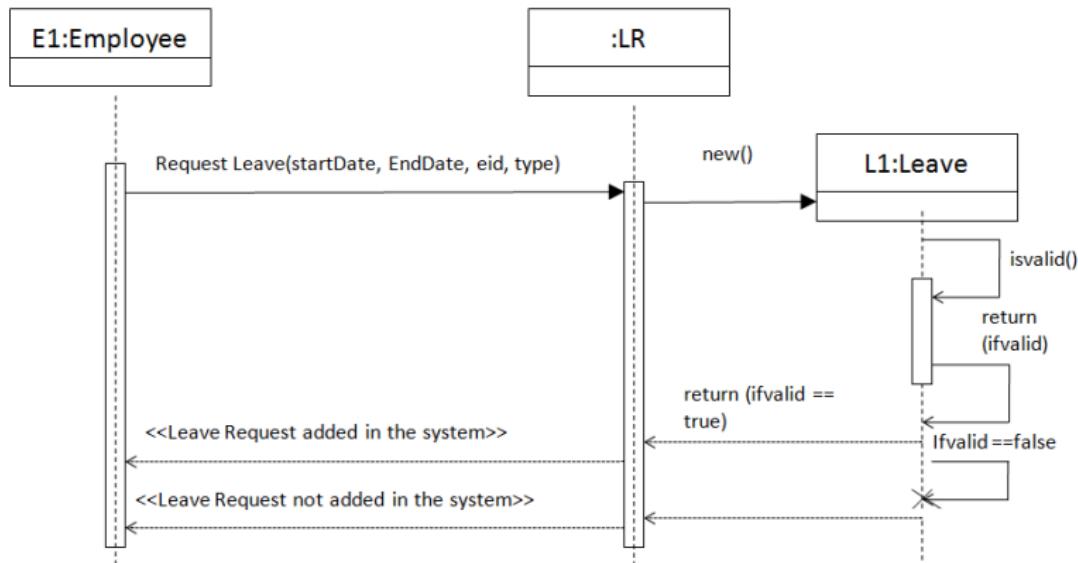
Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary





Sequence Diagram for LMS: Approve Leave

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

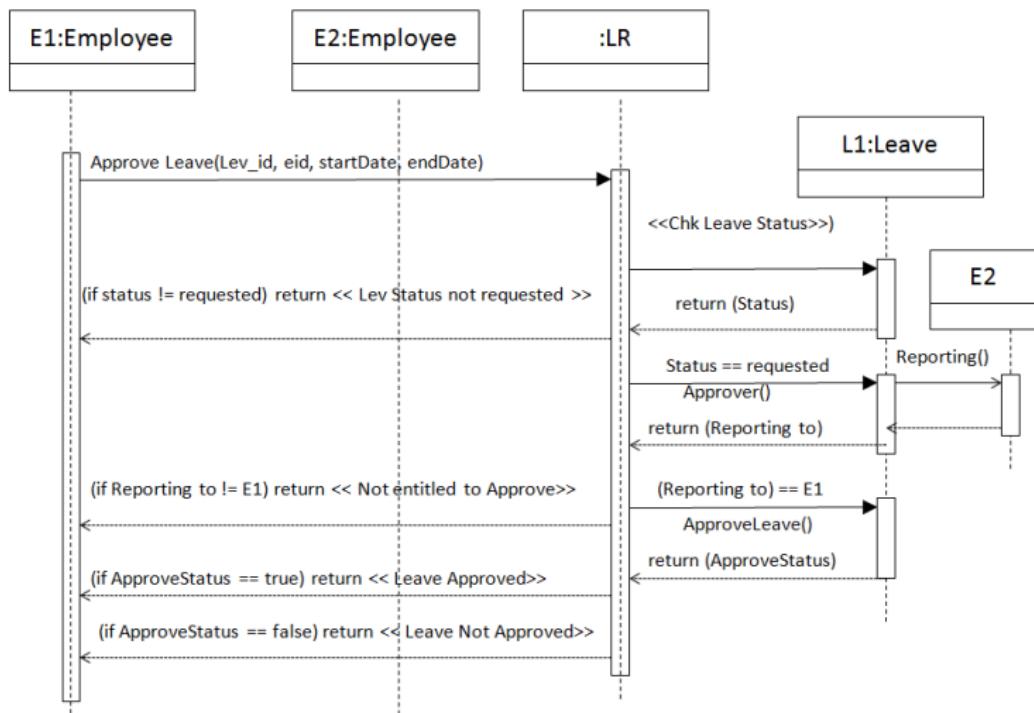
Interaction
Fragments

Occurrence
Execution

State Invariant
Interaction Use

LMS Sequence
Diagram

Summary





Module Summary

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary

- Discussed about interaction fragments in Sequence Diagrams
- Worked out the sequence diagram for two sample use-cases of LMS



Instructor and TAs

Module 30

Partha Pratim
Das

Objectives &
Outline

Sequence
Diagram

Interaction
Fragments
Occurrence
Execution
State Invariant
Interaction Use

LMS Sequence
Diagram

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

Module 31: Object Oriented Analysis & Design

Communication Diagram

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- Understanding Communication (Collaboration) Diagrams



Module Outline

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- What are Communication (Collaboration) Diagrams?
 - Frame
 - Lifeline
 - Messages
 - Examples



Client-Server Computing Model: RECAP (Modules 05, 11)

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- No object exists in isolation
- Objects are acted on and themselves act on other objects
- Leads to the **Client-Server Model** of computing where
 - Behavior is
 - Services provided by an object
 - Services are requested by
 - Sending Messages, Invoking Operations
 - In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



Communication (Collaboration) Diagrams in SDLC phases: RECAP (Module 22)

Module 31

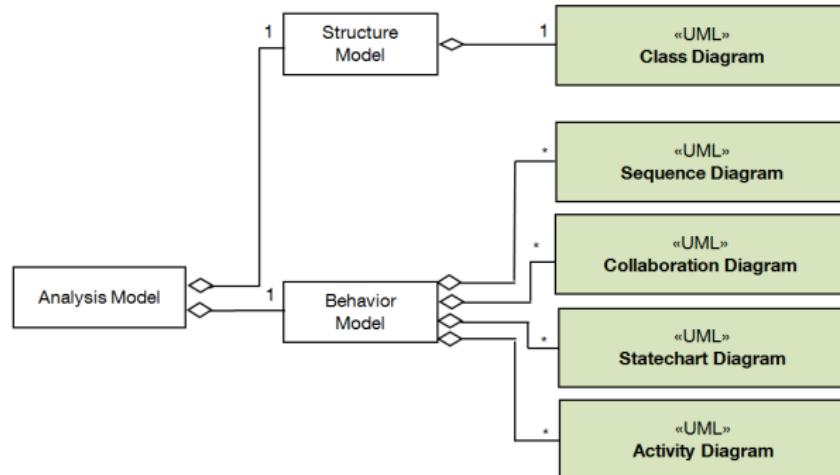
Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary



- In the **Analysis Phase** the problem domain is analyzed and refined from the **Requirements Phase**
- The behavior model of the system is hence understood in this phase
- Communication (Collaboration) diagrams is a result of the Analysis Phase



Communication (Collaboration) Diagrams in SDLC phases: RECAP (Module 22)

Module 31

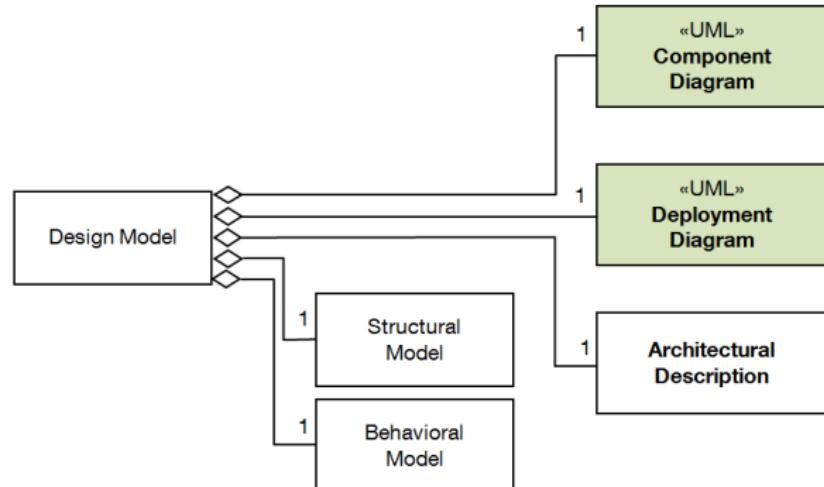
Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary



- Communication (Collaboration) diagram is included in the Behavioral Model
- It is further refined in the **Design Phase**



What are Communication Diagrams?

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- **Communication Diagram** (called **Collaboration Diagram** in UML 1.x) shows interactions between objects and / or parts (represented as lifelines) using sequenced messages in a free-form arrangement
- **Communication Diagram** is a **UML Behavior Diagram**
- **Communication Diagram** depicts the inter-object behavior of a system, ordered by space
- The major components of a **Communication Diagram** are:
 - Frames
 - Lifeline
 - Messages



Frames

Module 31

Partha Pratim
Das

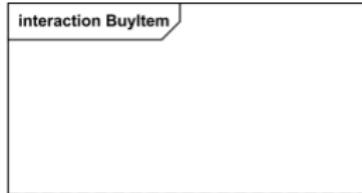
Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- Communication diagrams could be shown within a rectangular frame with the name in a compartment in the upper left corner



Interaction Frame for Communication Diagram
BuylItem



Sd Frame (short frame) for Communication
Diagram BuylItem

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Lifelines

Module 31

Partha Pratim
Das

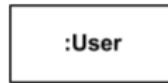
Objectives &
Outline

Communication
Diagrams

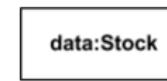
Frames
Lifelines
Messages
Examples

Summary

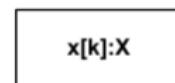
- Lifeline is a specialization of named element which represents an individual participant in the interaction
- A **Lifeline** is shown as a rectangle (corresponding to the **head** in sequence diagrams)
- Lifeline in sequence diagrams does have **tail** representing the line of life whereas **lifeline** in communication diagram has no line, just **head**
- The information identifying a lifeline is depicted as
ObjectName[selector]:ClassName



Anonymous lifeline of class User



Lifeline "data" of class Stock



Lifeline "x" of class X is selected
with selector [k]

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Messages

Module 31

Partha Pratim
Das

Objectives &
Outline

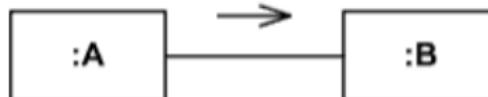
Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- Message in **Communication Diagram** is shown as a line with **sequence expression** and **arrow** above the line
- The arrow indicates direction of the communication

1.2.4 [s1.equals(s2)]: remove()



Instance of class A sends remove() message to
instance of B if s1 is equal to s2

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- The sequence expression is a dot separated list of sequence terms followed by a colon (":") and message name after that:
$$\text{sequence-expression} ::= \text{sequence-term}'\cdot'\dots'\cdot'\\ \text{message-name}$$
- Example: **3b.2.2:m5** : Sequence expression **3b.2.2** and message name **m5**
- Each Sequence term
$$\text{sequence-term} ::= [\text{integer}[\text{name}]] [\text{recurrence}]$$
- The **integer** represents the **sequential order** of the message within the next higher level of procedural calling

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression: Sequential Order

Module 31

Partha Pratim
Das

Objectives &
Outline

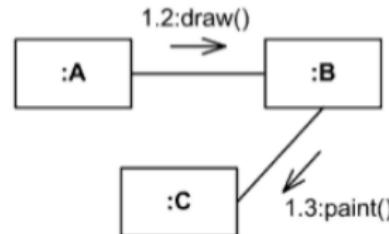
Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

● Example:

- message with sequence 2 follows message with sequence 1
- 2.1 follows 2
- 5.3 follows 5.2 within activation 5
- 1.2.4 follows message 1.2.3 within activation 1.2.



Instance of A sends draw() message to instance of B, and after that B sends paint() to C

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression: Concurrent Thread

Module 31

Partha Pratim
Das

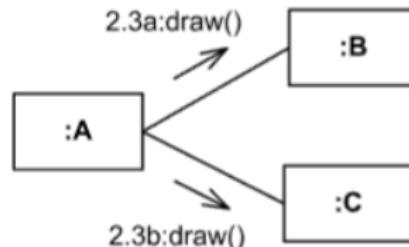
Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- The **name** represents a **concurrent thread** of control
- **Example:**
 - messages 2.3a and 2.3b are concurrent within activation 2.3
 - 1.1 follows 1a and 1b
 - 3a.2.1 and 3b.2.1 follow 3.2



Instance of A sends draw() messages concurrently to instance of B and to instance of C

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression: Guard

Module 31

Partha Pratim
Das

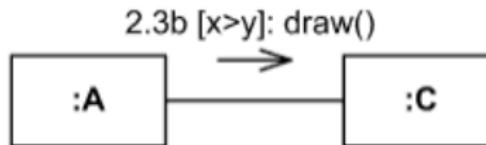
Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- A **guard** specifies condition for the message to be sent (executed) at the given nesting depth
- **Example:**
 - **2.3b [x>y]: draw()**: message draw() will be executed if x is greater than y
 - **1.1.1 [s1.equals(s2)]: remove()** – message remove() will be executed if s1 equals s2



Instance of class A will send message draw() to the instance of C, if $x > y$

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression: Recurrence and Iteration

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- The **recurrence** defines conditional or iterative execution of zero or more messages that are executed depending on the specified condition
`recurrence ::= branch | loop , branch ::= '[' guard ']`
- An **iteration** specifies a sequence of messages at the given nesting depth
- **Notation:**
 - $*$: *Messages Executed Sequentially*
 - $*||$: *Messages Executed Concurrently*

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression: Recurrence and Iteration

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

Example:

- **4.2c *[i=1..12]: search(t[i])** – search() will be executed 12 times, one after another
- **4.2c *||[i=1..12]: search(t[i])** – 12 search() messages will be sent concurrently
- **2.2 *: notify()** – message notify() will be repeated some unspecified number of times

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Sequence Expression: Recurrence and Iteration

Module 31

Partha Pratim
Das

Objectives &
Outline

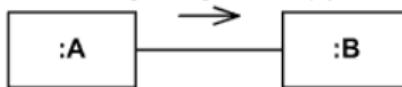
Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

Example:

1.2 *[k:1..n]: search(k)



Instance of class A will send search() message to instance of B n times, one by one

1.2 *||[k:1..n]: search(k)



Instance of class A will send n concurrent search() messages to instance of B

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Online Book Shop

Module 31

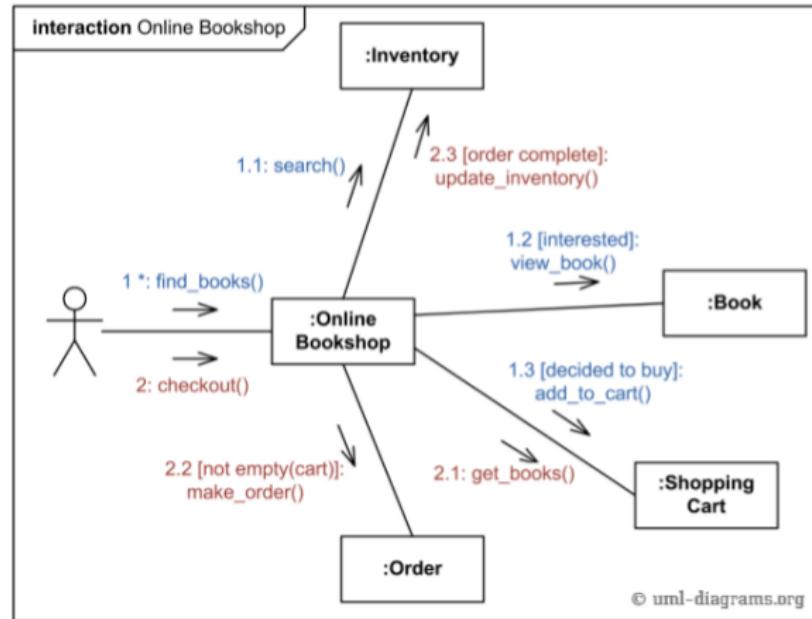
Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Online Book Shop - Annotated

Module 31

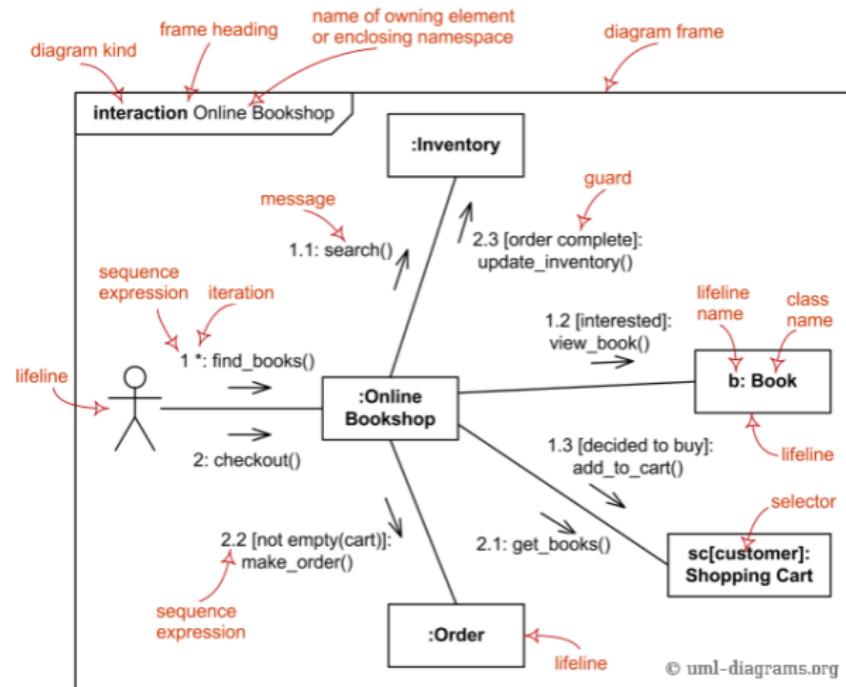
Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Order Management

Module 31

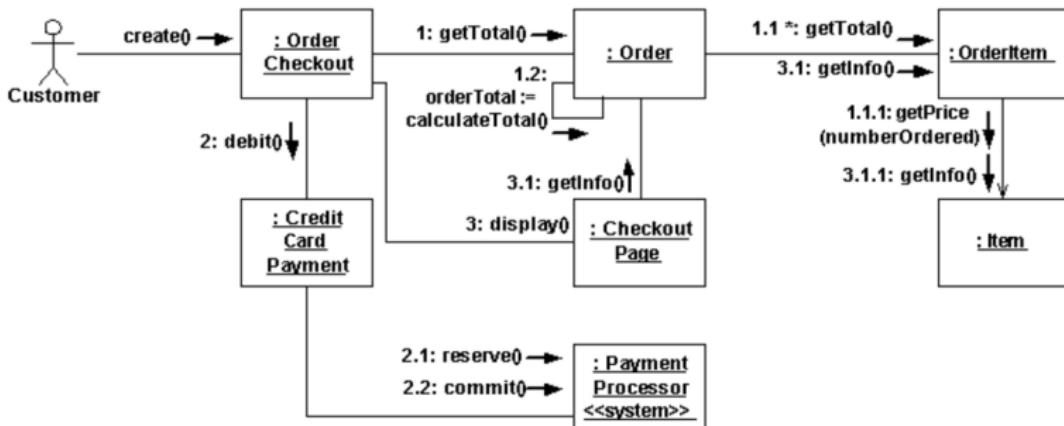
Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary



Source: <http://agilemodeling.com/style/collaborationDiagram.htm> (20-Aug-16)



Module Summary

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

- Communication Diagrams are introduced
- Various components of Communication Diagrams like Frames, Lifeline, Messages are discussed
- Examples are illustrated



Instructor and TAs

Module 31

Partha Pratim
Das

Objectives &
Outline

Communication
Diagrams

Frames
Lifelines
Messages
Examples

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

Module 32: Object Oriented Analysis & Design

Activity Diagrams: Part 1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

● Understanding Activity Diagrams



Module Outline

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

- What are Activity Diagrams?

- **Activity**
- **Partition**
- **Activity Edge**
- Control
- Objects
- Actions

- Activity Diagram for LMS



Client-Server Computing Model: RECAP (Modules 05, 11)

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

- No object exists in isolation
- Objects are acted on and themselves act on other objects
- Leads to the **Client-Server Model** of computing where
 - Behavior is
 - Services provided by an object
 - Services are requested by
 - Sending Messages, Invoking Operations
 - In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



Use cases represent major Activities of a System: RECAP (Modules 23)

Module 32

Partha Pratim
Das

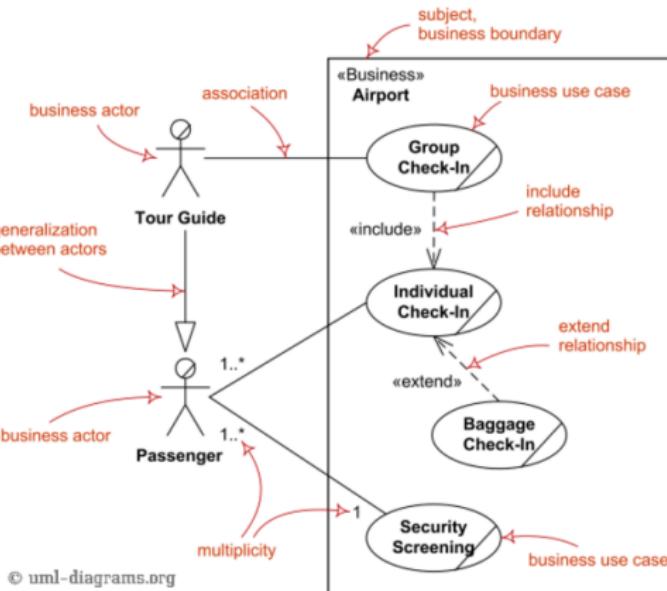
Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary



*Group Check-in, Individual Check-in, Baggage Check-in and Security Screening
are major activities of the Airport System*



Activity Diagrams in SDLC phases: RECAP (Module 22)

Module 32

Partha Pratim
Das

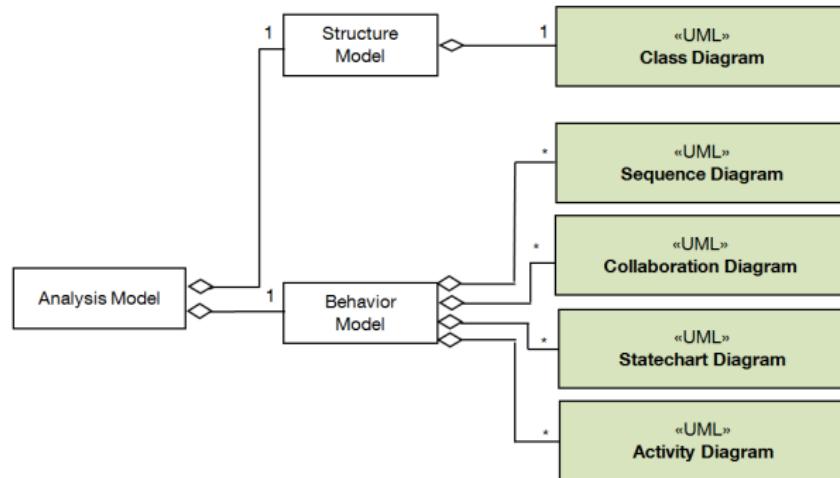
Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary



- In the **Analysis Phase** the problem domain is analyzed and refined from the **Requirements Phase**
- The behavior model of the system is hence understood in this phase
- **Activity diagram** is a result of the Analysis Phase



Activity Diagrams in SDLC phases: RECAP (Module 22)

Module 32

Partha Pratim
Das

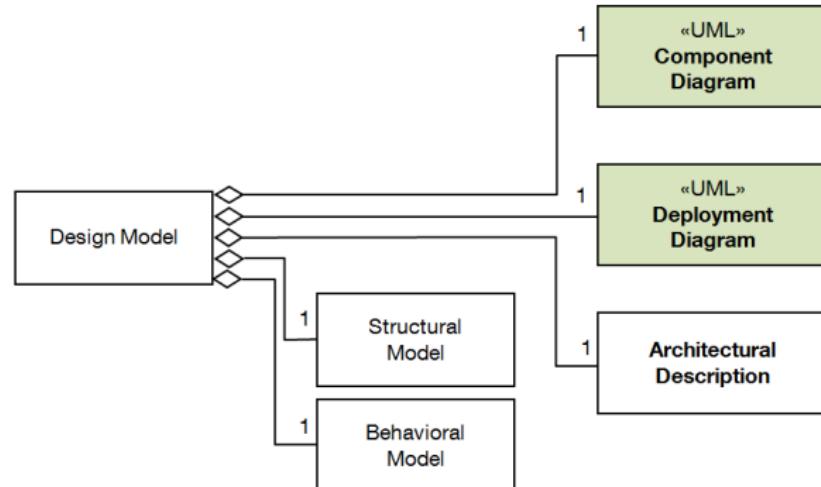
Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary



- Activity is included in the Behavioral Model
- It is further refined in the **Design Phase**



What are Activity Diagrams?

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

- **Activity Diagram** is a **UML behavior diagram** which shows **flow of control** or **object flow** with emphasis on the sequence and conditions of the flow
- Activity Diagrams resemble old-school flow-charts
- Typically used to model an algorithm (sequential as well as concurrent / parallel) or use-case realization



Blog Account Creation Process – Activity Diagram

Module 32

Partha Pratim
Das

Objectives &
Outline

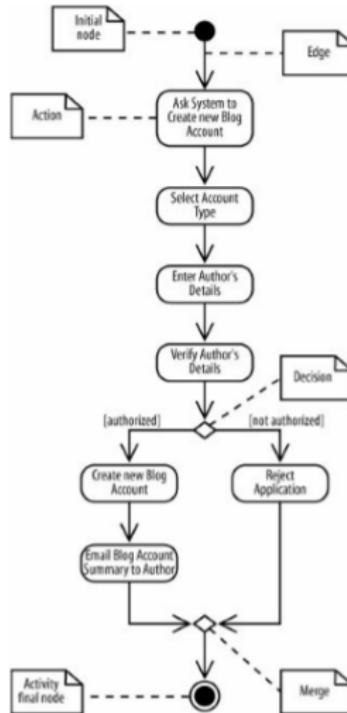
Activity
Diagrams

Activity
Activity
Partition

Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary





What are Activity Diagrams?

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

- The actions coordinated by activity models can be initiated because
 - other actions finish executing,
 - objects and data become available, or
 - some events external to the flow occur
- The major components of an [Activity Diagram](#) are:
 - Activity
 - Partition
 - Activity Edge
 - Control
 - Objects
 - Actions



Activity

Module 32

Partha Pratim
Das

Objectives &
Outline

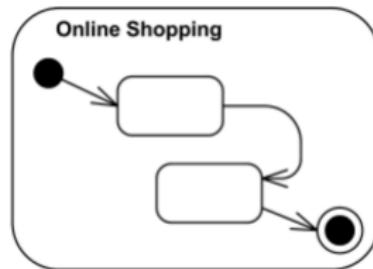
Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

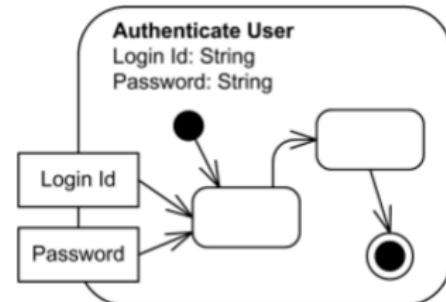
Example

Summary

- **Activity** is a parameterized **behavior** represented as coordinated flow of **actions**
- It is denoted as round-cornered rectangle with activity name in the upper left corner containing the nodes and edges
- Activity parameters are displayed on the border as: **parameter-name: parameter-type**



Online Shopping activity



Authenticate User activity with two parameters -
Login Id and Password

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Activity

Module 32

Partha Pratim
Das

Objectives &
Outline

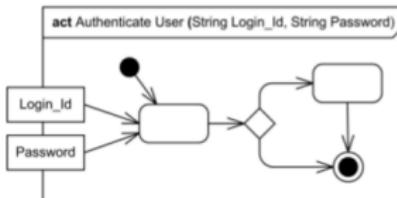
Activity
Diagrams

Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

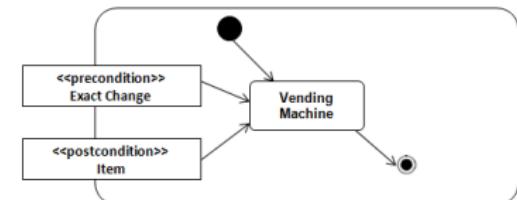
Example

Summary

- As a **behavior** activity could have pre- and post-condition constraints, shown with the keywords `<<precondition>>` and `<<postcondition>>` respectively



Authenticate User activity frame with two
parameters - Login Id and Password



Vending machine activity with precondition and
post condition

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Activity Partition

Module 32

Partha Pratim
Das

Objectives &
Outline

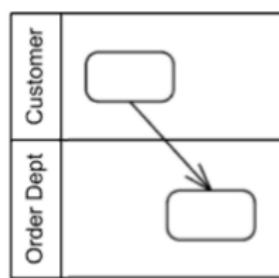
Activity
Diagrams

Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

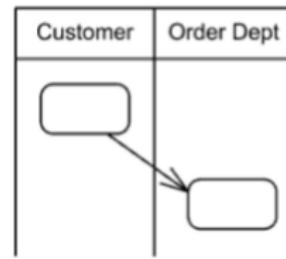
Example

Summary

- An **activity partition** is **activity group** for actions that have some common characteristic
- Partitions often correspond to **organizational units** or **business actors** in a **business model**
- **Activity partition** is shown with a **swimlane notation** - with two, usually parallel lines, either horizontal or vertical, with the partition name in a box at one end



Activity partitions Customer and Order Dept as
horizontal swimlanes



Activity partitions Customer and Order Dept as
vertical swimlanes

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Activity Partition

Module 32

Partha Pratim
Das

Objectives &
Outline

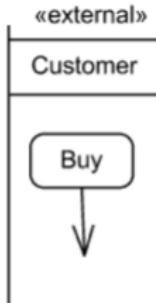
Activity
Diagrams

Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

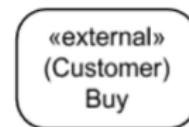
Summary

- Partition could represent an external entity to which the partitioning structure does not apply, labeled with keyword `<>external<>`



Buy action occurs in external partition Customer -

Swimlane notation



Buy action occurs in external partition Customer -
Activity notation

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Activity Edge

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

- Activity Edge is an abstract class for the directed connections along which tokens or data objects flow between activity nodes
- It includes
 - control edges
 - object flow edges



Activity edge connects Fill Order and Review Order



Activity edge "updated" connects two nodes

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Activity Edge

Module 32

Partha Pratim
Das

Objectives &
Outline

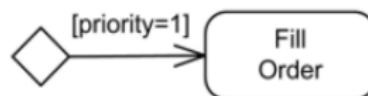
Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

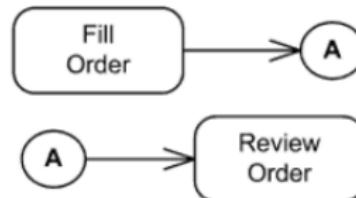
Summary

- Activity edge can have a guard – specification evaluated at run-time to determine if the edge can be traversed
- The guard of the activity edge is shown in square brackets that contain the guard**



Fill Order when priority is 1

- An activity edge can be notated using a connector, which is a small circle with a name inside



Connector A connects two edges between Fill Order and Review Order

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Object Flow Edge

Module 32

Partha Pratim
Das

Objectives &
Outline

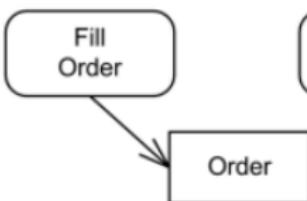
Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

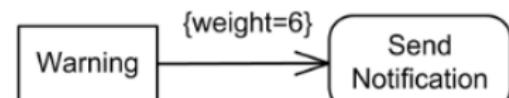
Example

Summary

- Object flow edges are activity edges used to show data flow of object and data tokens between action nodes
- The **weight** attribute dictates the minimum number of tokens that must traverse the edge at the same time



Object flow of Orders between Fill Order and
Review Order actions



Send Notification when number of Warnings
reaches 6

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Interrupting Edge

Module 32

Partha Pratim
Das

Objectives &
Outline

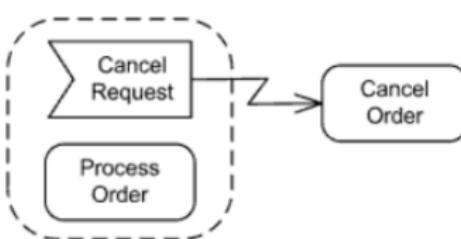
Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

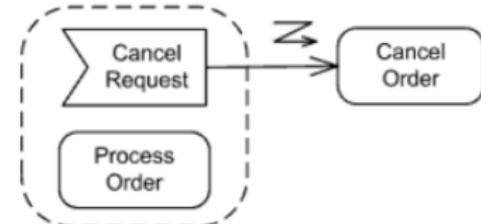
Example

Summary

- Interrupting edge is activity edge expressing interruption for regions having interruptions
- It is rendered as a lightning-bolt or zigzag adornment on a straight line



Cancel Request signal causes interruption
resulting in Cancel Order



Cancel Request signal causes interruption
resulting in Cancel Order

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Online Shopping – Activity Diagram

Module 32

Partha Pratim
Das

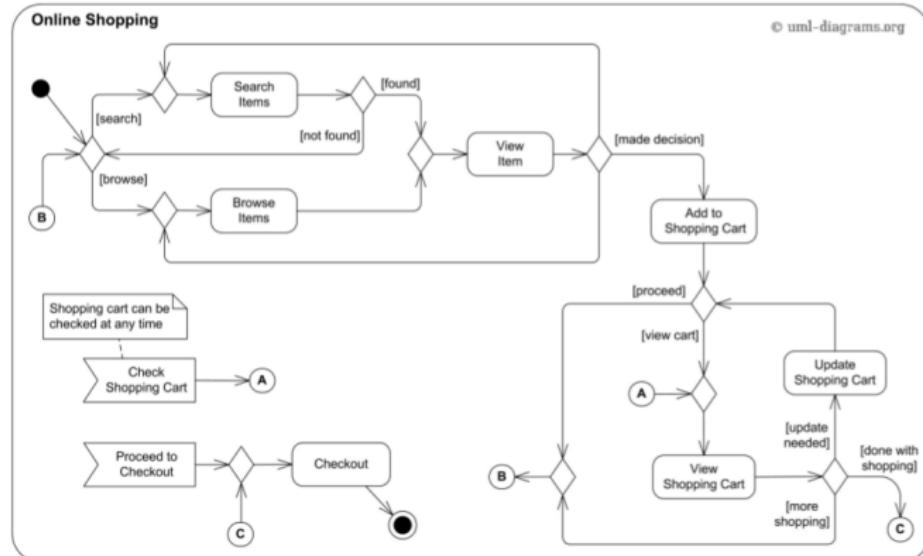
Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Ticket Vending Machine – Activity Diagram

Module 32

Partha Pratim
Das

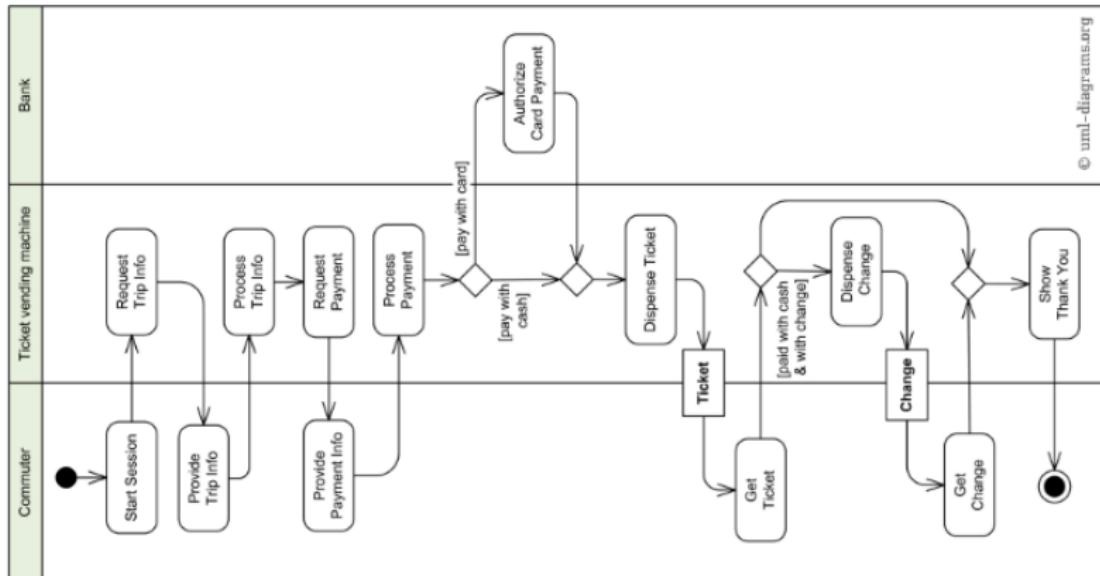
Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Module Summary

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

- Activity Diagrams are introduced
- Various components of Activity Diagram like Activity, partition, and edge are discussed
- Examples are illustrated



Instructor and TAs

Module 32

Partha Pratim
Das

Objectives &
Outline

Activity
Diagrams

Activity
Activity
Partition
Activity Edge
Object Flow
Edge
Interrupting
Edge

Example

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

Module 33: Object Oriented Analysis & Design

Activity Diagrams: Part 2

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

- Understanding Control features of Activity Diagrams



Module Outline

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

- What are Activity Diagrams?

- Activity
- Partition
- Activity Edge
- Control
- Objects
- Actions

- Activity Diagram for LMS



Activity Diagram: RECAP (Modules 32)

Module 33

Partha Pratim
Das

Objectives &
Outline

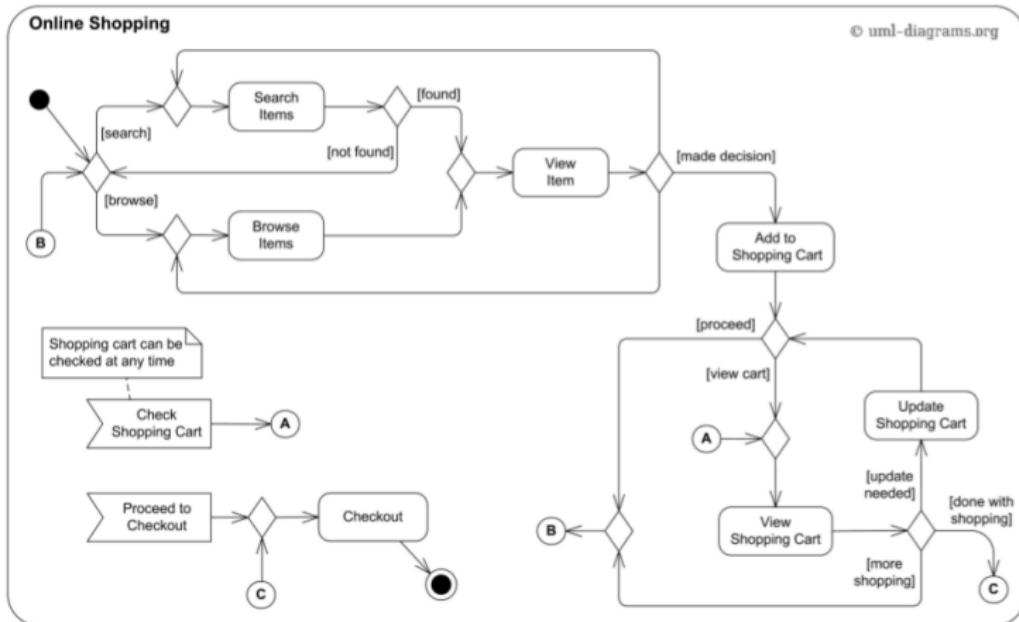
Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)
NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das



Controls

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

- Control node is an activity node used to coordinate the flows between other nodes
- It includes:
 - Initial Node
 - Flow Final Node
 - Activity Final Node
 - Decision Node
 - Merge Node
 - Fork Node
 - Join Node



Activity Control Node Overview

Module 33

Partha Pratim
Das

Objectives &
Outline

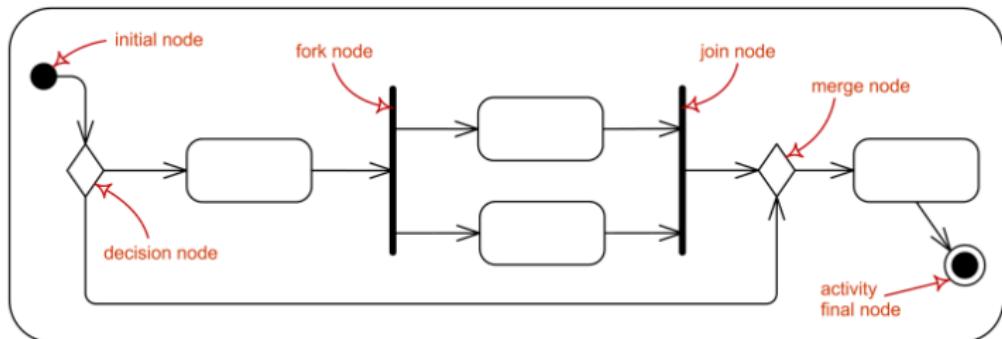
Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary



Activity control nodes overview

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Initial, Flow Final, and Activity Final Node

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

Initial node is a control node at which flow starts when the activity is invoked

Notation: Small solid circle

Flow final node is a control final node that terminates a flow

Notation: Small circle with X inside



Activity initial node



Flow final node

Activity final node is a control final node that stops all flows in an activity

Notation: Solid circle with a hollow circle inside



Activity final node

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Decision Node

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

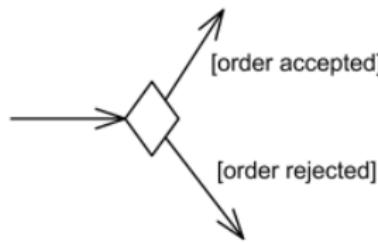
Example

Summary

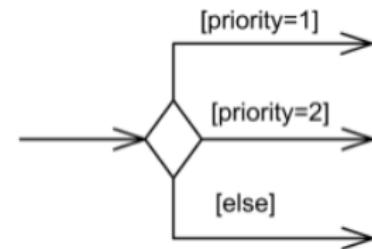
Decision node is a control node that accepts tokens on one or two incoming edges and selects one outgoing edge from one or more outgoing flows

Guards define which outgoing edge will be traversed

Notation: Diamond-shaped symbol



Decision node with two outgoing edges with guards



Decision node with three outgoing edges and [else] guard

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Decision Node

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

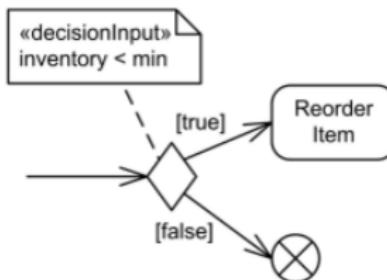
Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

Tokens (Incoming activity) is passed to the **Decision input behavior** before guards are evaluated on the outgoing edges

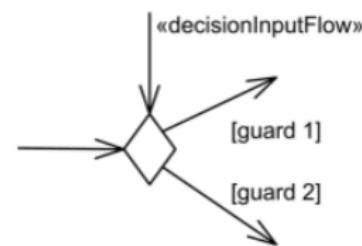
Notation: `<<decisionInput>>`, note



Decision node with decision input behavior

Tokens (Incoming activity) of the **Decision input flow** are made available to the guards before evaluation

Notation: `<<decisionInputFlow>>`



Decision node with decision input flow

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Merge Node

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node

Merge Node

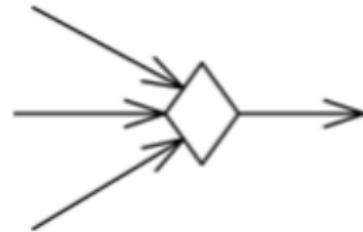
Fork Node

Join Node

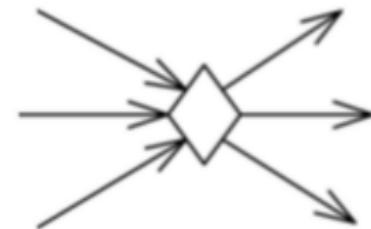
Example

Summary

- As a **Merge node** is a control node that brings together multiple incoming alternate flows to accept single outgoing flows (Control and Object Flows)
- Merge Node is non-blocking**
- Notation: Diamond-shaped symbol with two or more edges entering it and a single activity edge leaving it*



Merge node with three incoming edges and
a single outgoing edge



Merge node and decision node combined using the
same symbol

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Decision-Merge Nodes: Example

Module 33

Partha Pratim
Das

Objectives &
Outline

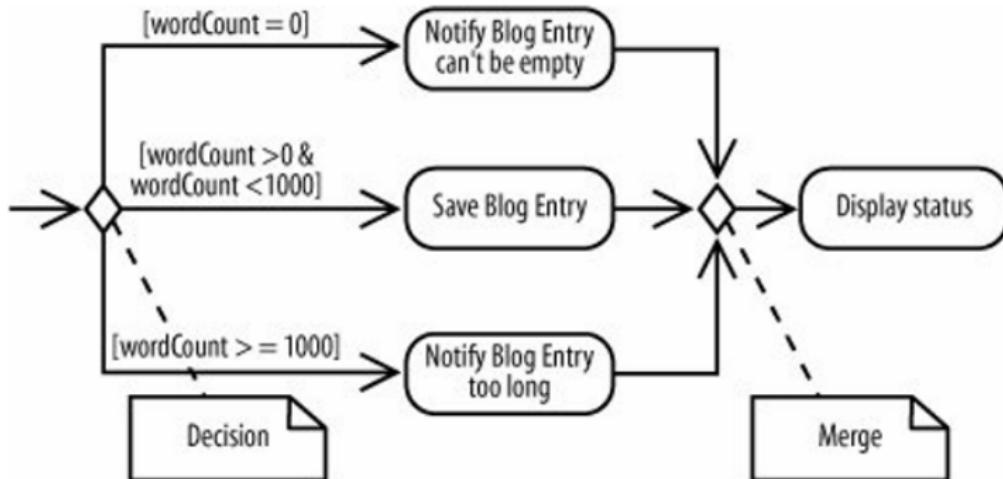
Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary





Concurrency: RECAP (Module 10)

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

- **Concurrency** is *the property that distinguishes an active object from one that is not active*
- Allows multiple tasks to execute, interact and collaborate at the same time to achieve the global functionality
- Concurrency is critical for *Client-Server Model* of computation



Fork Node

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

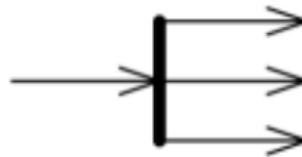
Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

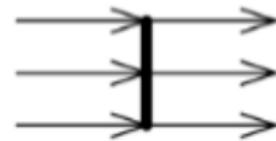
Example

Summary

- **Fork node** is a control node that has one incoming edge and multiple outgoing edges and is used to split incoming flow into multiple **concurrent** flows
- *Notation: Line segment with a single activity edge entering it, and two or more edges leaving it*



Fork node with a single activity edge
entering it, and three edges leaving it



Combined join node and fork node

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Join Node

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

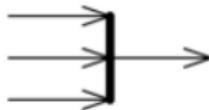
Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

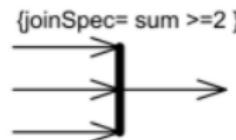
- **Join node** is a control node that has multiple incoming edges and one outgoing edge and is used to synchronize incoming **concurrent** flows
- **Join Node is blocking**
- **Notation:** *line segment with several activity edges entering it, and only one edge leaving it*



Join node with three activity edges entering it, and a single edge leaving it



Combined join node and fork node



Join node with join specification shown in curly braces

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Fork-Join Nodes: Example

Module 33

Partha Pratim
Das

Objectives &
Outline

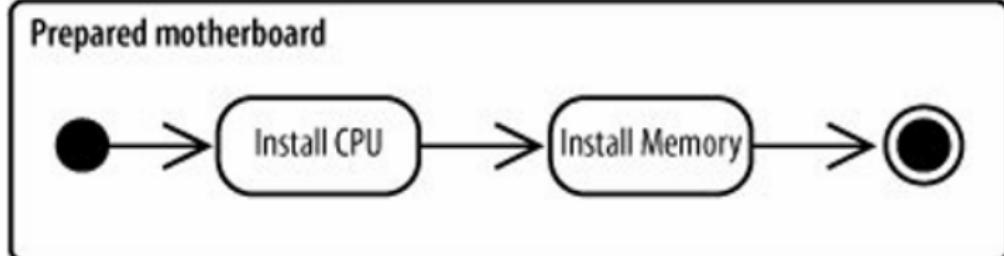
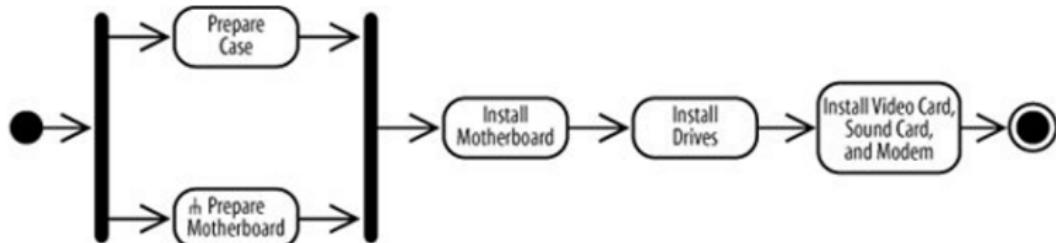
Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary





Online Shopping – Activity Diagram

Module 33

Partha Pratim
Das

Objectives &
Outline

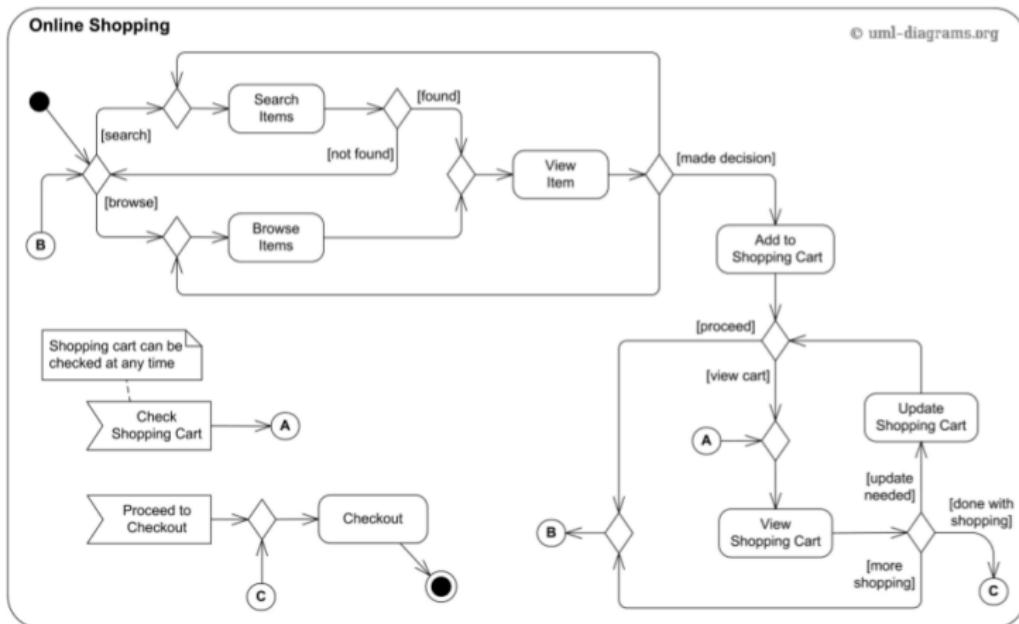
Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)
NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das

16



Product Process – Activity Diagram

Module 33

Partha Pratim
Das

Objectives &
Outline

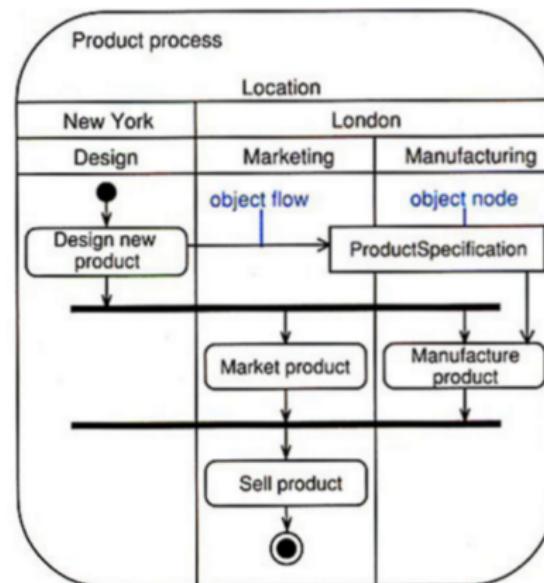
Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary



Source: url: <http://pja.mykhi.org/mgr/blokowe/INN/sorcersoft.org/io/uml/ActivityDiagrams.html>

(22-Aug-16)

NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das

17



Process Order – Activity Diagram

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow,
Final, Activity
Final Node

Decision Node

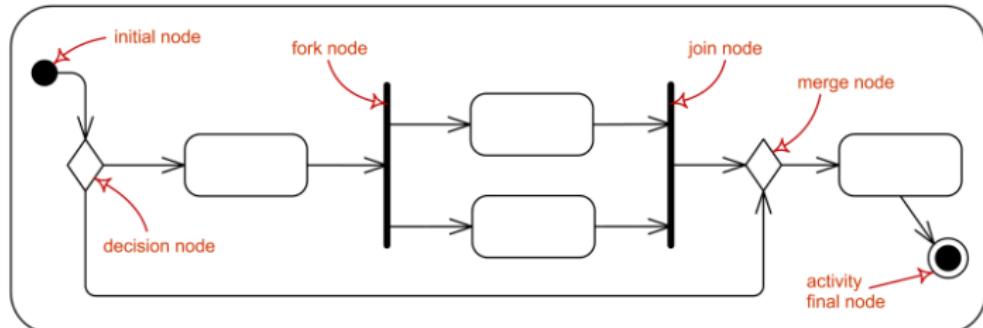
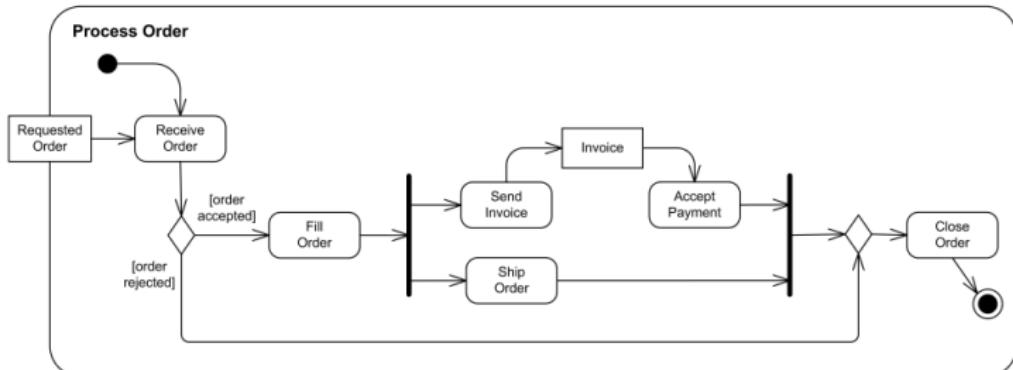
Merge Node

Fork Node

Join Node

Example

Summary



Source: http://www.inf.ed.ac.uk/teaching/courses/seoc/2009_2010/notes/1_notes.pdf(20-Aug-16)



Module Summary

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

- Control Node of Activity Discussed
- Examples are illustrated



Instructor and TAs

Module 33

Partha Pratim
Das

Objectives &
Outline

Controls

Initial, Flow
Final, Activity
Final Node

Decision Node
Merge Node
Fork Node
Join Node

Example

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

Module 34: Object Oriented Analysis & Design

Activity Diagrams: Part 3

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

- Understanding the various features of Activity Diagrams
- Deriving the Activity Diagram for LMS



Module Outline

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

- What are Activity Diagrams?

- Activity
- Partition
- Activity Edge
- Control
- Objects
- Actions

- Activity Diagram for LMS



Activity Diagram: RECAP (Modules 31)

Module 34

Partha Pratim
Das

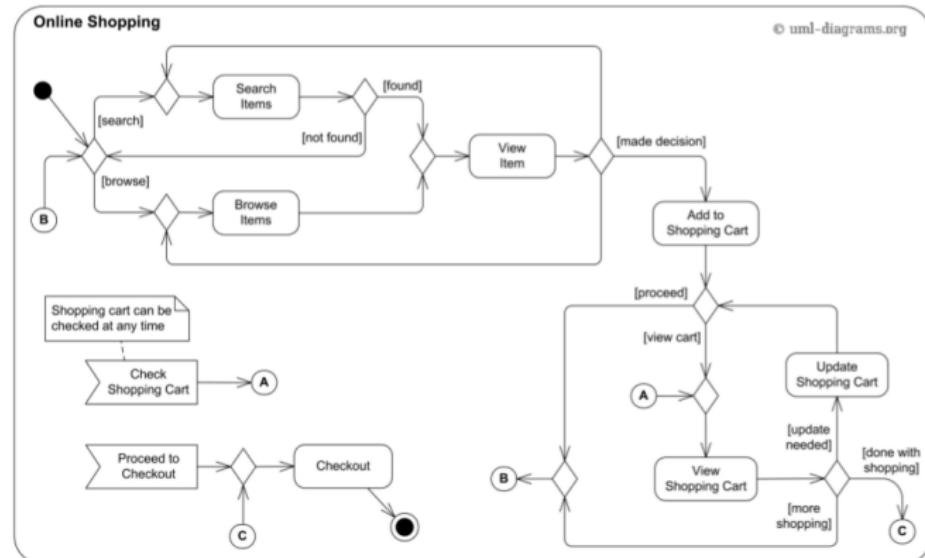
Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Objects

Module 34

Partha Pratim
Das

Objectives &
Outline

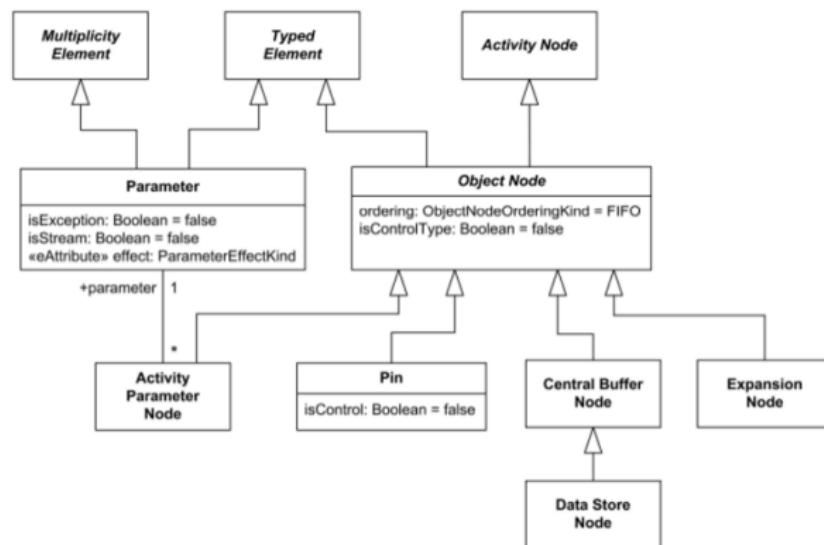
Objects

Actions

Activity
Diagram for
LMS

Summary

- An object is an abstract activity node that is used to define object flows in an activity
- Object nodes include **pin**, **central buffer**, **parameter**, and **expansion nodes**



Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Object Node

Module 34

Partha Pratim
Das

Objectives &
Outline

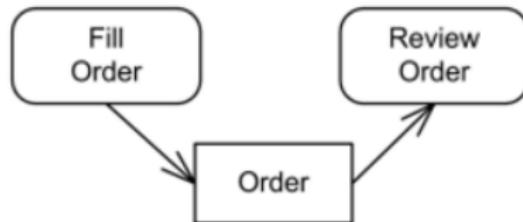
Objects

Actions

Activity
Diagram for
LMS

Summary

- An object node is an abstract activity node that is used to define object flow in an activity
- It represents a particular state of a class.
- Object nodes are notated as rectangles



Object flow of Orders between Fill Order and Review Order actions

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Pin

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

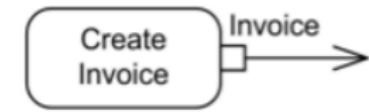
Activity
Diagram for
LMS

Summary

- A pin is an object node for inputs and outputs to actions
- Pin is usually shown as a small rectangle attached to the action rectangle



Item is input pin to the Add to Shopping Cart
action



Invoice is output pin from the Create Invoice
action

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Central Buffer

Module 34

Partha Pratim
Das

Objectives &
Outline

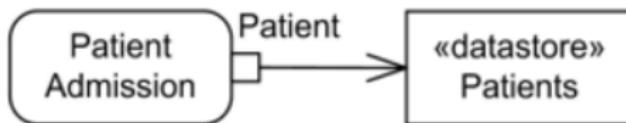
Objects

Actions

Activity
Diagram for
LMS

Summary

- A central buffer node is an object node for managing flows from multiple sources and destinations
- A **data store** is a central buffer node for non-transient information
- All incoming tokens are stored by the data store



Incoming Patient token is stored by the Patients data store

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Process Order – Activity Diagram

Module 34

Partha Pratim
Das

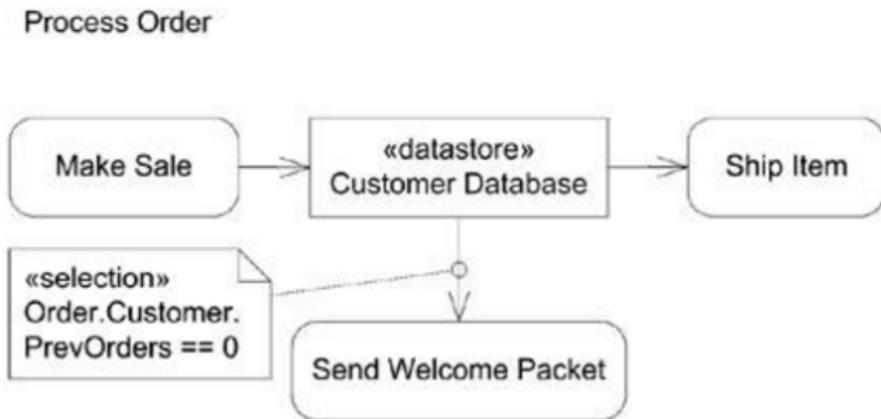
Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary



Source: url: <http://flylib.com/books/en/4.282.1.66/1/> (22-Aug-16)



Distribute Cars - Activity Diagram

Module 34

Partha Pratim
Das

Objectives &
Outline

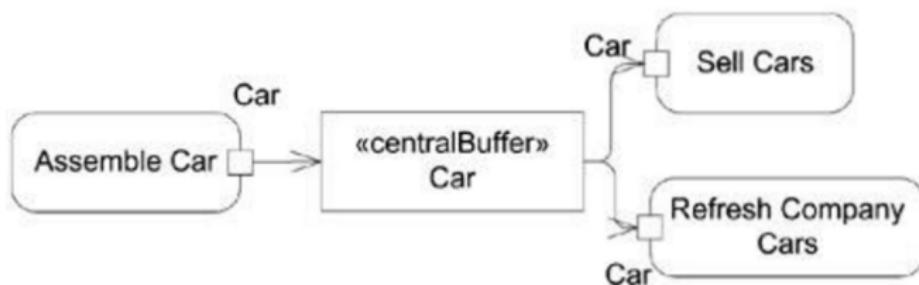
Objects

Actions

Activity
Diagram for
LMS

Summary

Distribute Cars



Source: url: <http://flylib.com/books/en/4.282.1.66/1/> (22-Aug-16)



Actions

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

- Action is a named element which represents a single atomic step within activity that is not further decomposed within the activity
- Action could also be expressed in some application-dependent action language

Process
Order

The Process Order action.

```
for (Account a: accounts)  
    a.verifyBalance();  
end_for
```

Example of action expressed with tool-dependent
action language.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Actions

Module 34

Partha Pratim
Das

Objectives &
Outline

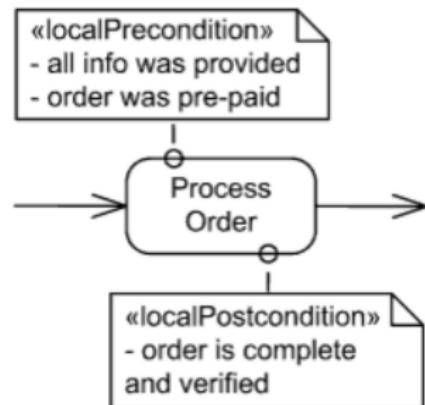
Objects

Actions

Activity
Diagram for
LMS

Summary

- Action can have **local pre and post conditions** attached as note



Local pre- and post-conditions shown as notes attached to Process Order action

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Dispense Drinks

Module 34

Partha Pratim
Das

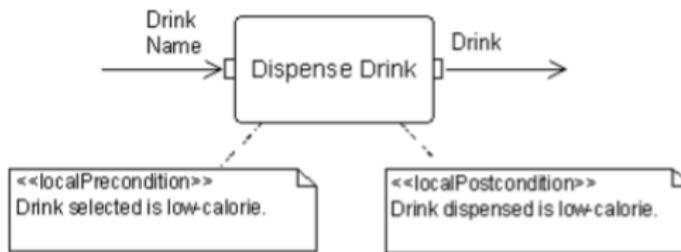
Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary



Source: url: http://www.jot.fm/issues/issue_2003_09/column4/ (22-Aug-16)



Objects and Actions: Example

Module 34

Partha Pratim
Das

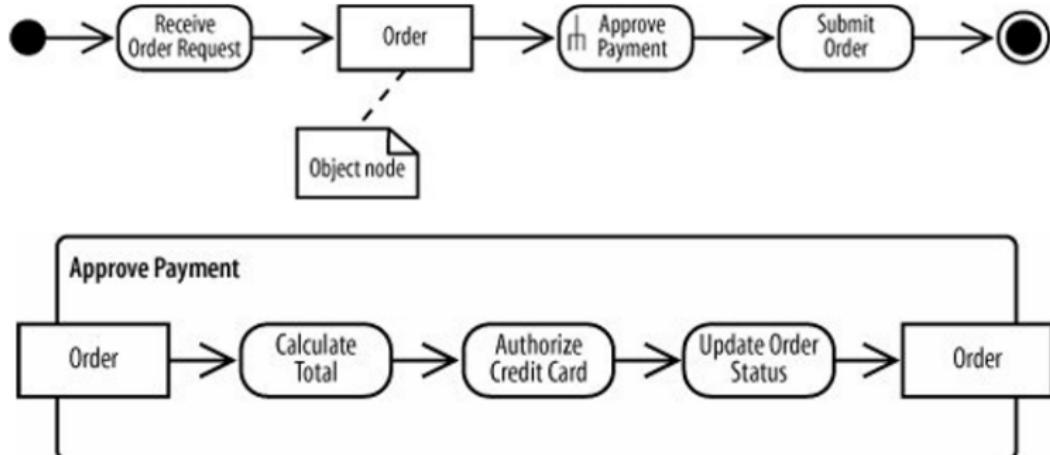
Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary





Objects and Actions: Example

Module 34

Partha Pratim
Das

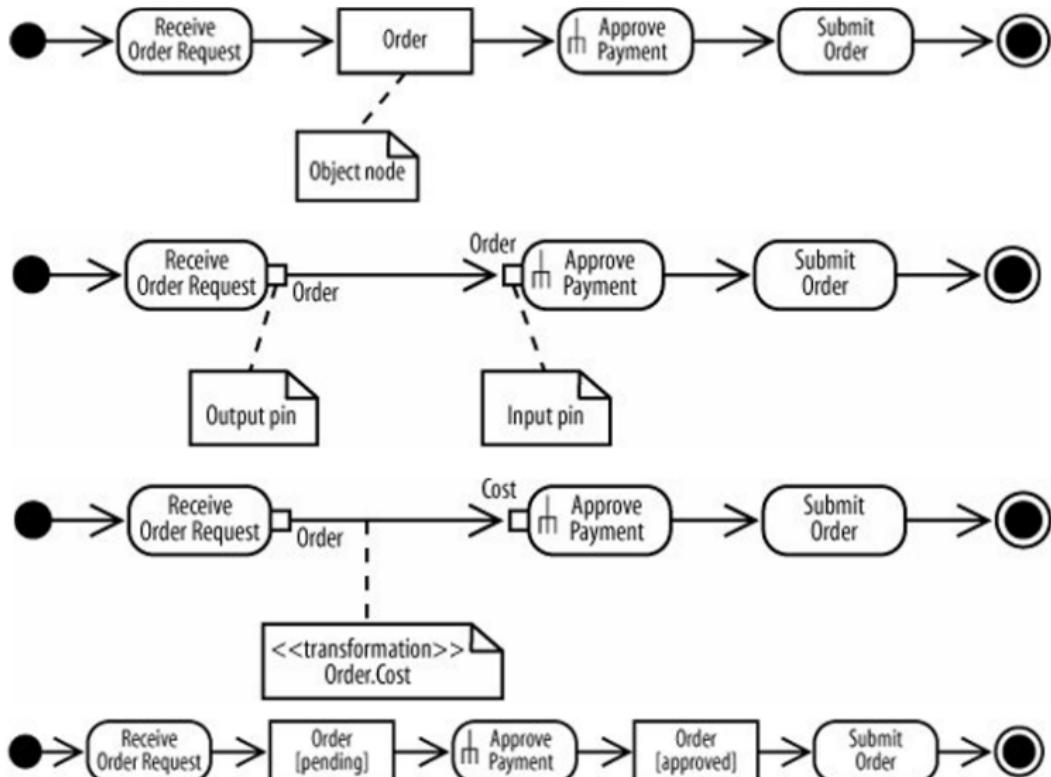
Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary





Activity Diagram for LMS

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

We will now derive the Activity Diagram for LMS. The steps are:

- Identify the activities and partitions
- Identify the control nodes, object nodes and actions of the activities
- Derive the final Activity Diagram



Activities of LMS

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

The messages for the major activities of LMS are given below:

- **Request Leave**
- **Approve Leave**
- **Revoke Leave**
- **Cancel Leave**
- **Avail Leave**
- **Export Leave**



Activity Partition of LMS

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

Shown below are two major activity partition of LMS

Employee {Abstract}	Leave {Abstract}



Activity Diagram of LMS

Module 34

Partha Pratim
Das

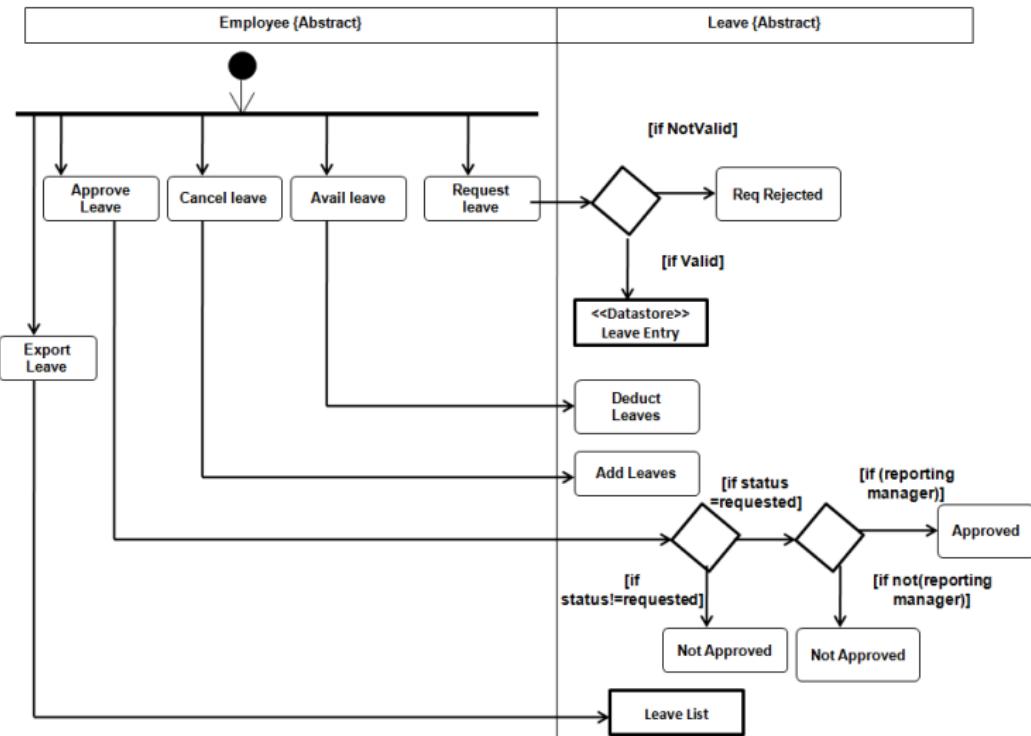
Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary





Module Summary

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

- Object and Action Feature of Activity Diagram discussed
- Examples are illustrated
- Activity Diagram for LMS is derived



Instructor and TAs

Module 34

Partha Pratim
Das

Objectives &
Outline

Objects

Actions

Activity
Diagram for
LMS

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

Interaction Use

Duration

Constraint

Time

Constraint

Example

Summary

Module 35: Object Oriented Analysis & Design

Interaction Overview Diagram

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example

Summary

- Understanding Interaction Overview Diagrams



Module Outline

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example

Summary

• What are Interaction Overview Diagrams?

- Frames
- Elements of Activity Diagram
- Elements of Interaction
 - Interaction
 - Interaction use
 - Duration constraint
 - Time constraint



Client-Server Computing Model: RECAP (Modules 05, 11)

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

Interaction Use

Duration
Constraint

Time
Constraint

Example

Summary

- No object exists in isolation
- Objects are acted on and themselves act on other objects
- Leads to the **Client-Server Model** of computing where
 - Behavior is
 - Services provided by an object
 - Services are requested by
 - Sending Messages, Invoking Operations
 - In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



Interaction Overview Diagrams in SDLC phases: RECAP (Module 22)

Module 35

Partha Pratim
Das

Objectives &
Outline

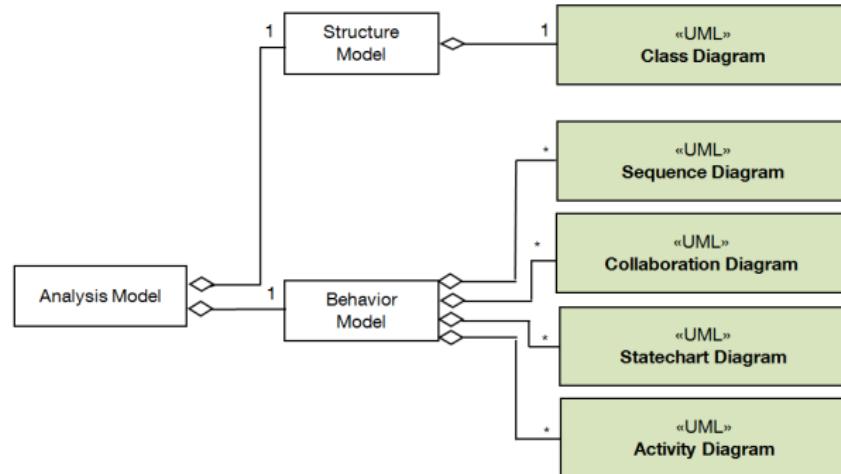
Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example
Summary



- In the **Analysis Phase** the problem domain is analyzed and refined from the **Requirements Phase**
- The behavior model of the system is hence understood in this phase
- Interaction Overview diagrams represents **the overall flow of control consisting of various interaction diagram fragments**



Interaction Overview Diagrams in SDLC phases: RECAP (Module 22)

Module 35

Partha Pratim
Das

Objectives &
Outline

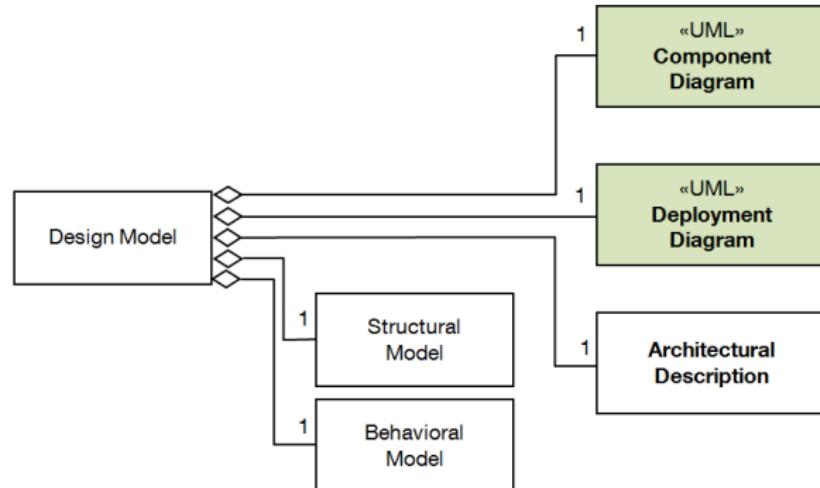
Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example
Summary



- Interaction Overview diagram is included in the Behavioral Model
- It is further refined in the **Design Phase**



What are Interaction Overview Diagrams?

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use

Duration
Constraint
Time
Constraint

Example

Summary

- Interaction overview diagrams provide overview of the flow of control where nodes of the flow are interactions or interaction uses
- UML interaction overview diagram combines elements from activity and interaction (sequence majorly) diagrams
- The major components of a Interaction overview diagrams Diagram are:
 - Frames
 - Elements of Activity Diagram
 - Elements of Interaction
 - Interaction
 - Interaction use
 - Duration constraint
 - Time constraint



Interaction Overview Diagram – Overview

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

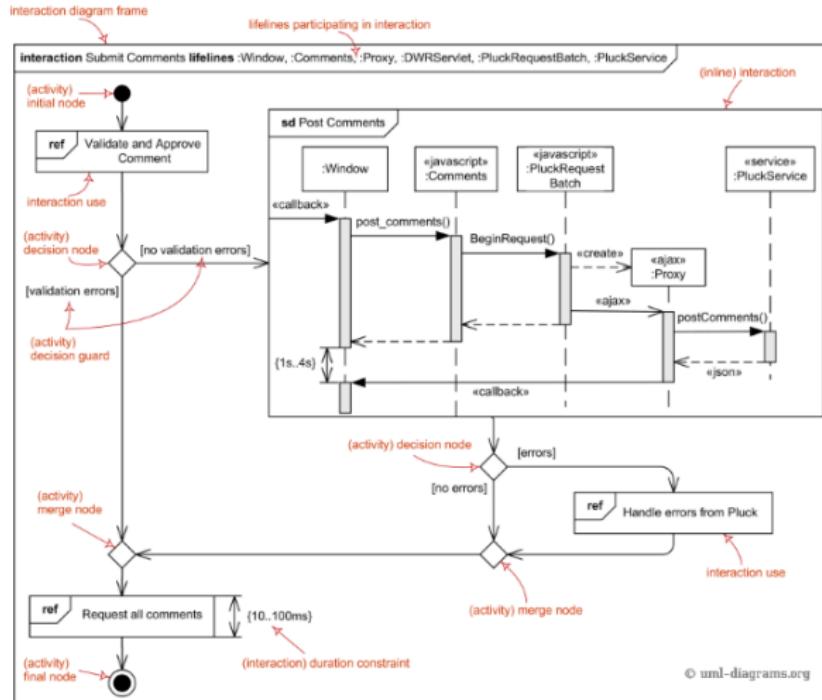
Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint

Time
Constraint

Example

Summary



UML interaction overview diagram combines elements from activity and interaction diagrams.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Frames

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use

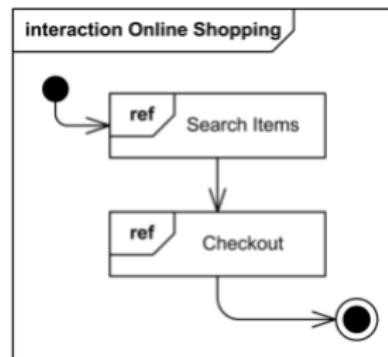
Duration
Constraint

Time
Constraint

Example

Summary

- Interaction overview diagrams are framed by the same kind of frame that encloses other forms of interaction diagrams
- Notation: A rectangular frame around the diagram with a name in a compartment in the upper left corner*
Interaction kind is interaction or sd (abbreviated form)



Interaction overview diagram Online Shopping

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Elements of Activity Diagram

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

Interaction Use
Duration
Constraint

Time
Constraint

Example

Summary

The following elements of the activity diagrams could be used on interaction overview diagrams

- initial node
- flow final node
- activity final node
- decision node
- merge node
- fork node
- join node

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Enroll in Seminar

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

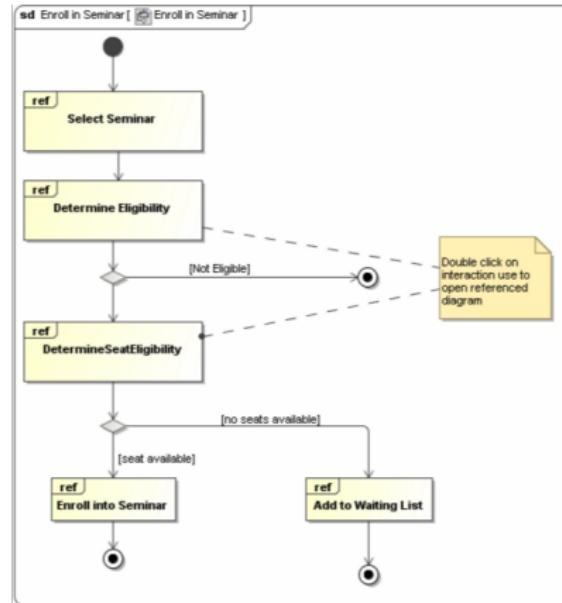
Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint

Time
Constraint

Example

Summary



Interaction Overview Diagram containing elements of Activity Diagram

Source: [url: http://docs.nomagic.com/display/MD183/Interaction+Overview+diagram?](http://docs.nomagic.com/display/MD183/Interaction+Overview+diagram?)

preview=/6130044/6130045/interaction_overview_diagram.png(20-Aug-16)



Elements of Interaction Diagram

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example

Summary

The following elements of the Interaction (majorly Sequence) diagrams could be used on interaction overview diagrams

- **interaction**
- **interaction use**
- **duration constraint**
- **time constraint**

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Interaction

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

Interaction Use

Duration

Constraint

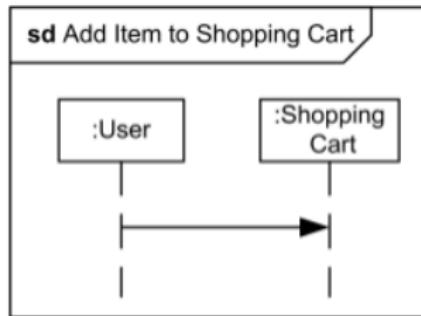
Time

Constraint

Example

Summary

- An interaction diagram of any kind may appear inline as an invocation action
- The inline interaction diagrams may be either anonymous or named



Interaction Add Item to Shopping Cart may appear inline on some interaction overview diagram

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Interaction Use

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

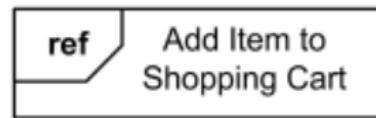
Interaction
Interaction Use

Duration
Constraint
Time
Constraint

Example

Summary

- An interaction use may appear as an invocation action



Interaction use Add Item to Shopping Cart may appear on some interaction overview diagram

- The inline interaction will have arguments (if any) of the reference replaced with parameters

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Duration Constraint

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

Interaction Use

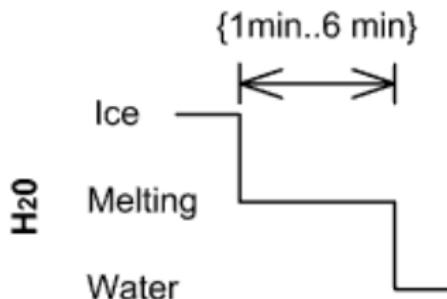
Duration
Constraint

Time
Constraint

Example

Summary

- Duration constraint is an **interval constraint** that refers to a **duration interval**
- The duration interval is duration used to determine whether the constraint is satisfied



Ice should melt into water in 1 to 6 minutes

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Time Constraint

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

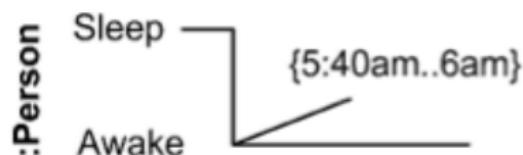
Interaction Use
Duration
Constraint

Time
Constraint

Example

Summary

- Time constraint is an **interval constraint** that refers to a **time interval**.
- The time interval is time expression used to determine whether the constraint is satisfied.



Person should wake up between 5:40 am and 6 am

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (20-Aug-16)



Access Control: Example

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

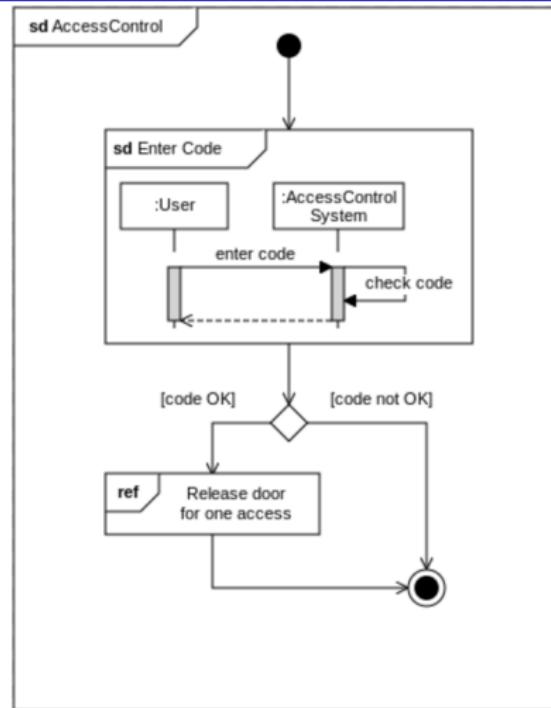
Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example

Summary



Source: url:https://en.wikipedia.org/wiki/Interaction_overview_diagram#/media/File:Uml-Iod-Diagram1.svg(20-Aug-16)



System Authentication: Example

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

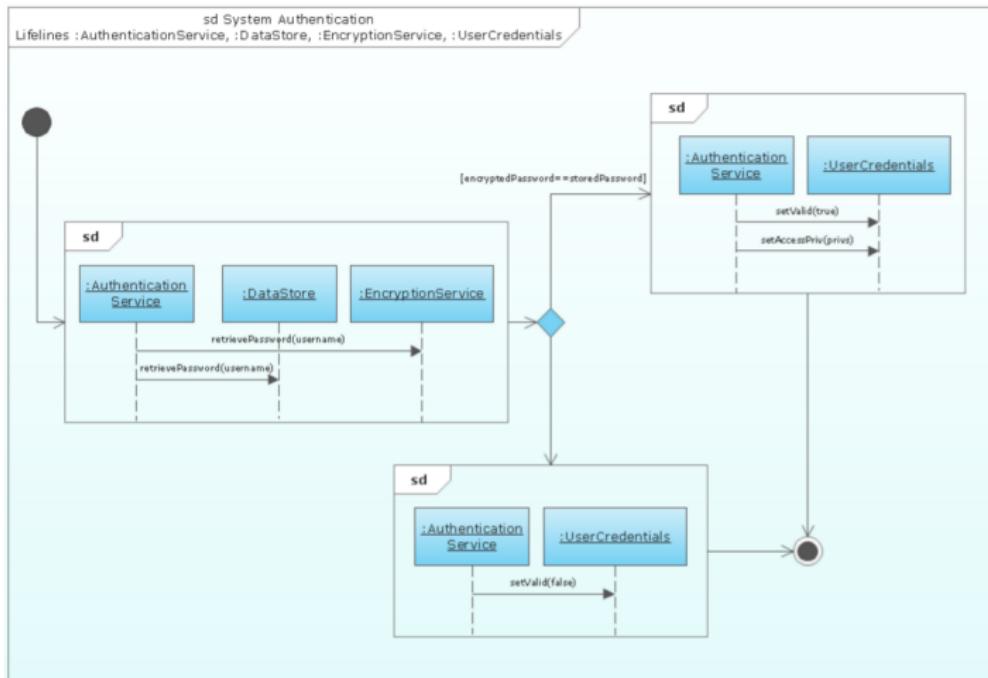
Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint
Time
Constraint

Example

Summary



Source: [url: http://www.conceptdraw.com/How-To-Guide/picture/diagrams-software-tools-for-design-uml-interaction-overview-diagram/UML-interaction-overview-diagram-System-authentication.png\(20-Aug-16\)](http://www.conceptdraw.com/How-To-Guide/picture/diagrams-software-tools-for-design-uml-interaction-overview-diagram/UML-interaction-overview-diagram-System-authentication.png)



Module Summary

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames

Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction

Interaction Use

Duration

Constraint

Time

Constraint

Example

Summary

- Interaction Overview Diagrams are introduced
- Various components of Interaction Overview Diagrams like Frames, Interaction fragments are discussed
- Examples are illustrated



Instructor and TAs

Module 35

Partha Pratim
Das

Objectives &
Outline

Interaction
Overview
Diagrams

Frames
Elements of
Activity Diagram

Elements of
Interaction
Diagram

Interaction
Interaction Use
Duration
Constraint

Time
Constraint

Example

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

Module 36: Object Oriented Analysis & Design

State Machine Diagram: Part 1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

● Understanding State Machine Diagrams



Module Outline

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex

Behavioral
State

Module
Summary

- What are State Machine Diagrams?
 - Behavioral State Machine
 - Vertex
 - Behavioral State
 - Pseudostate
 - Final State
 - Behavioral Transition
 - Protocol State Machine
 - State
 - Transition
- State Machine Diagram for LMS



STATE of an Object: RECAP (Module 11)

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

The **State** of an Object is a combination of values for its properties:

- Consider a Complex number having two properties:

Complex

- re: double // Real Part
- im: double // Imaginary Part

- Its states are possible pairs of values of re and im. For example:

- (2.3, 7.4)
- (-17.3627, 12.9)
- (29.0, -11.11)



BEHAVIOR of an Object: RECAP (Module 11)

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

The **Behavior** of an Object is the collection of its **operations** which may or may not change the **state** of an object:

- Consider the Complex number objects:

Stack
<ul style="list-style-type: none">- store: char[]- marker: int
<ul style="list-style-type: none">+ Push(int): void+ Pop(): void+ Top(): char+ Empty(): bool+ Print(): void

- It supports 4 common stack operations
- In addition, there will be Constructor, Destructor etc.
- Print() is not a usual stack operation – included for debugging and illustration
- Stack cannot be used to Search() an item!



Client-Server Computing Model: RECAP (Modules 05, 11)

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- No object exists in isolation
- Objects are acted on and themselves act on other objects
- Leads to the **Client-Server Model** of computing where
 - Behavior is
 - Services provided by an object
 - Services are requested by
 - Sending Messages, Invoking Operations
 - In Client-Server View
 - Clients request for Services
 - Servers provide Services
 - Contract between client and server ensures correctness



State Machine Diagrams in SDLC phases: RECAP (Module 22)

Module 36

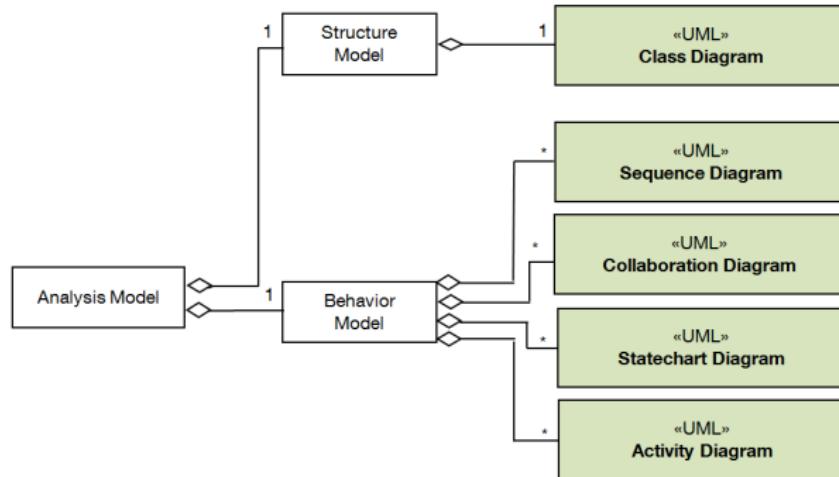
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary



- In the **Analysis Phase** the problem domain is analyzed and refined from the **Requirements Phase**
- The behavior model of the system is hence understood in this phase
- State Machine diagrams is a major result of the Analysis Phase



State Machine Diagrams in SDLC phases: RECAP (Module 22)

Module 36

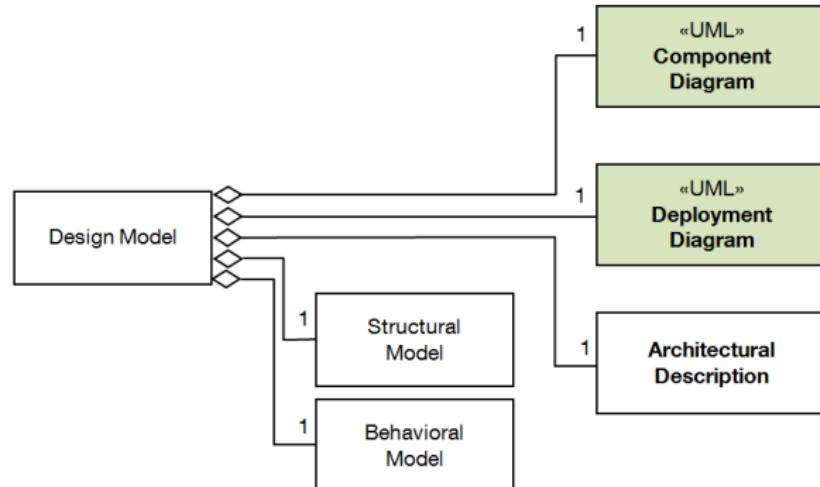
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary



- State Machine diagram is included in the Behavioral Model
- It is further refined in the **Design Phase**



What are State Machine Diagrams?

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- State machine diagram is a behavior diagram which shows discrete behavior of a part of designed system through finite state transitions
 - Behavioral State Machine
 - Protocol State Machine
- A state machine diagram mainly consists of **States** and **Transitions**



Behavioral State Machine

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine

Vertex
Behavioral
State

Module
Summary

Behavioral state machine

- is a specialization of **behavior** and is used to specify discrete behavior of a part of designed system through finite state transitions
- is modeled as a traversal of a graph of **state** nodes connected with **transitions**
- could be owned by **behaviored class** which is called its **context**
- The context defines which **signal** and **call triggers** are defined for a state machine
- may have an associated **behavioral feature (specification)** and be the **method** of this behavioral feature

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Behavioral State Machine

Module 36

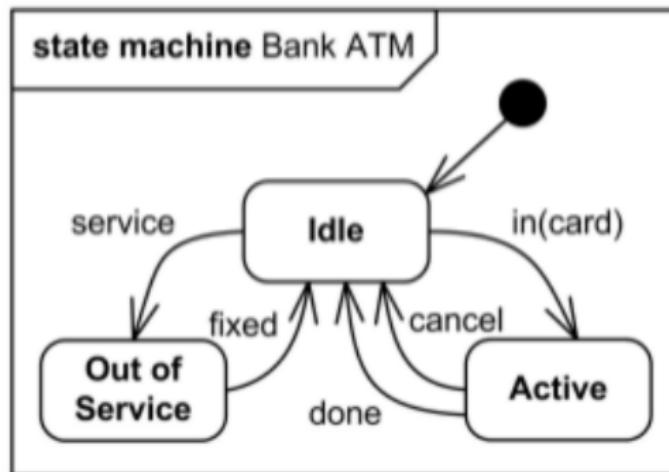
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary



High level behavioral state machine for bank ATM

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Behavioral State Machine

Module 36

Partha Pratim
Das

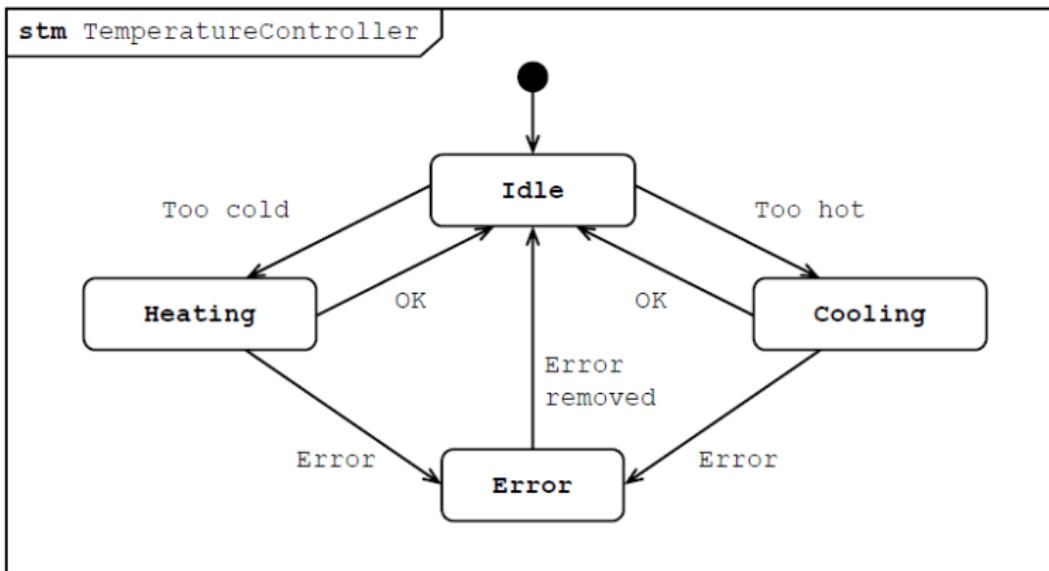
Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine

Vertex
Behavioral
State

Module
Summary



Behavioral state machine for Temperature Controller



A behavioral State Machine consists of **Vertex** and **Behavioral Transition**

- **Vertex** is named element which is an abstraction of a node in a state machine graph
 - In general, it can be the source or destination of any number of **transitions**
 - Subclasses of Vertex are:
 - Behavioral State
 - Pseudostate
- **State** is a **vertex** which models a situation during which some (usually implicit) invariant condition holds

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Behavioral State

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- Behavioral State models a situation during which some (usually implicit) invariant condition holds
- The invariant may
 - represent a static situation such as an object waiting for some external event to occur
 - model dynamic conditions such as the process of performing some behavior
- The various kinds of states are:
 - Simple State
 - Composite State
 - Submachine State

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Simple State

Module 36

Partha Pratim
Das

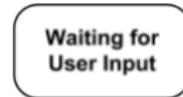
Objectives &
Outline

State Machine
Diagrams

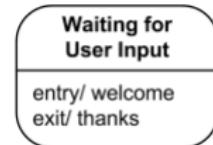
Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- A **simple state** is a state that does not have substates
- *Notation: Rectangle with rounded corners and the state name inside the rectangle*
- **State may have compartments**
 - name: (optional) name of the state. *State name can be optional*
 - internal activities: (**do**) activities (behaviors) while in state, (**entry**) and (**exit**) activities
 - internal transitions: a list of internal transitions, where each item has the form as described for **trigger**



Simple state Waiting for Customer Input



Simple state Waiting for Customer Input with
name and internal activities compartment

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Water Phase Management

Module 36

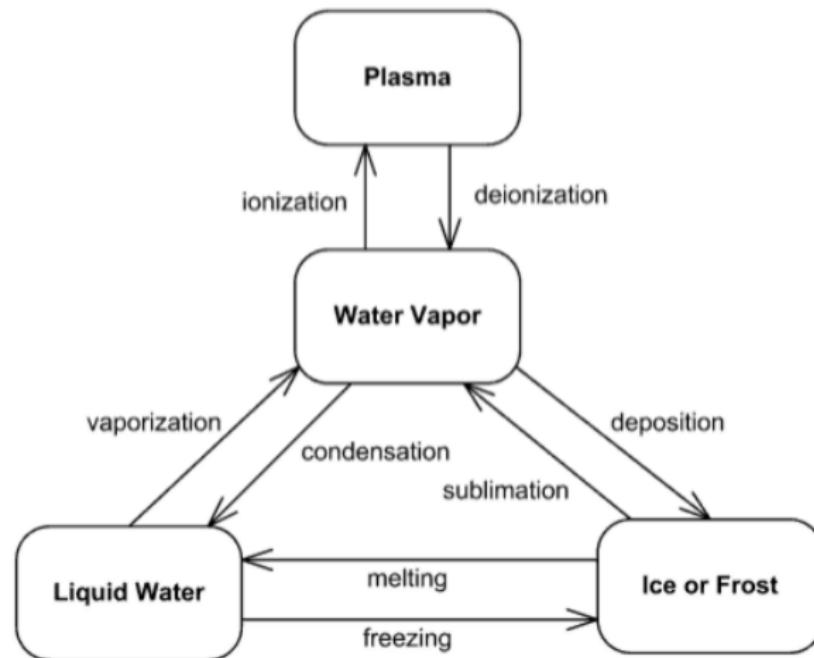
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary



Source: <http://agilemodeling.com/style/collaborationDiagram.htm> (20-Aug-16)



Behavioral State Machine

Module 36

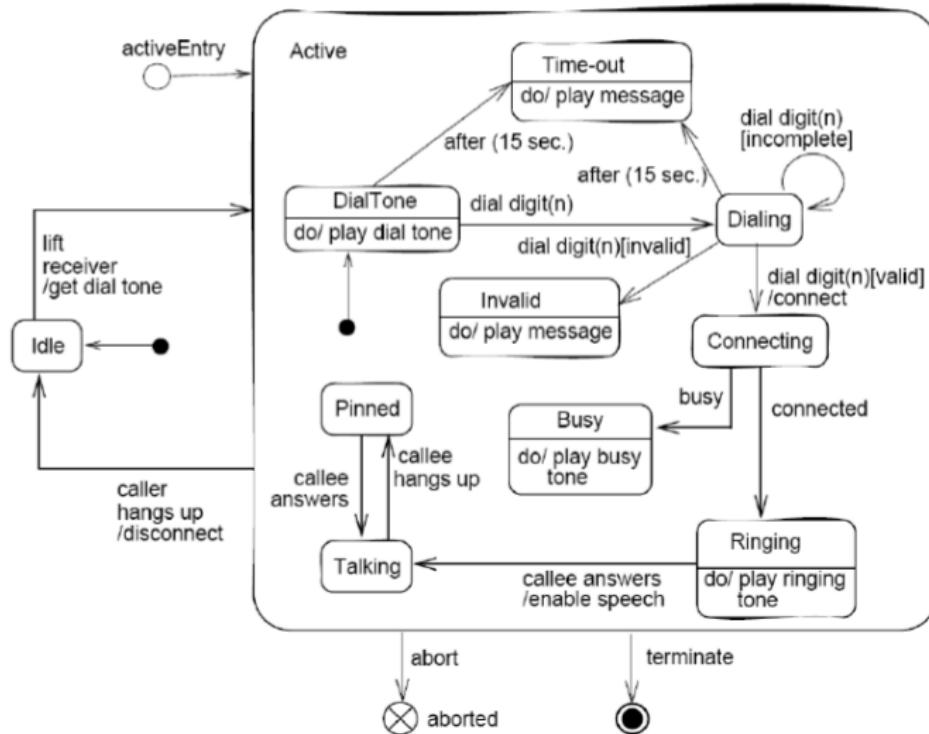
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary



Behavioral state machine for Dialing a Phone



Composite State

Module 36

Partha Pratim
Das

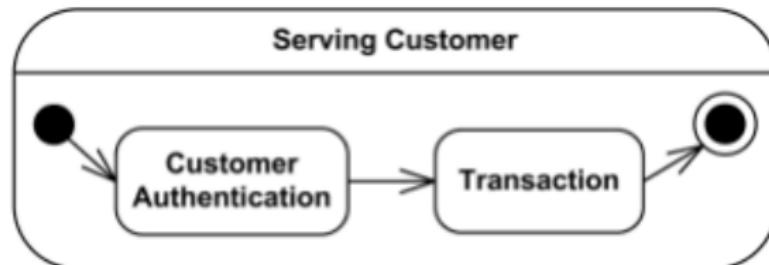
Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- A **composite state** is defined as state that has substates (nested states)
- Substates could be **sequential (disjoint)** or **concurrent (orthogonal)**
- A composite state can have one or more **regions**
- A **region** contains states and transitions
- **Simple composite state** contains just one region



Simple composite state Serving Customer has two substates

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Composite State

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- **Orthogonal composite state** has more than one regions
- Any state enclosed within a region of a composite state is called a substate of that composite state
- A composite state has an additional **decomposition compartment** apart from the initial 3 compartments
- **Decomposition compartment** shows composition structure of the state consisting of regions, states, and transitions



Composite state Serving Customer with decomposition hidden

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Composite State

Module 36

Partha Pratim
Das

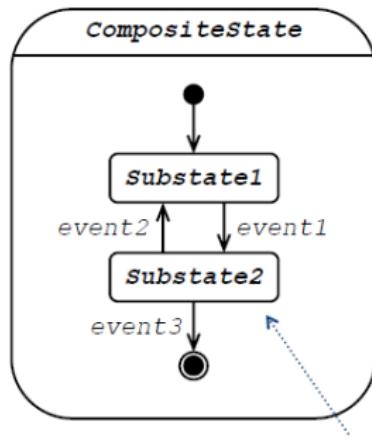
Objectives &
Outline

State Machine
Diagrams

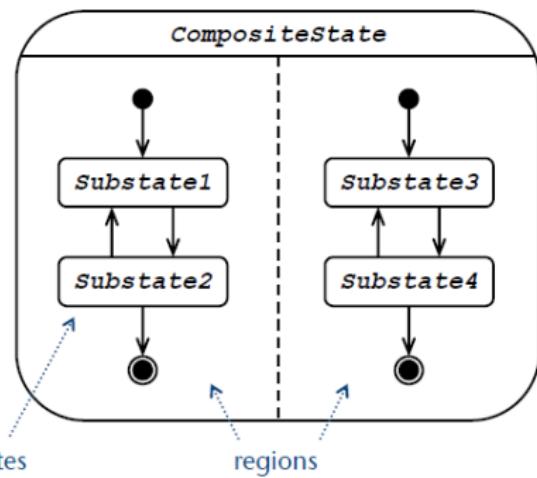
Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

Non-orthogonal
composite state:



Orthogonal composite state:





Composite State

Module 36

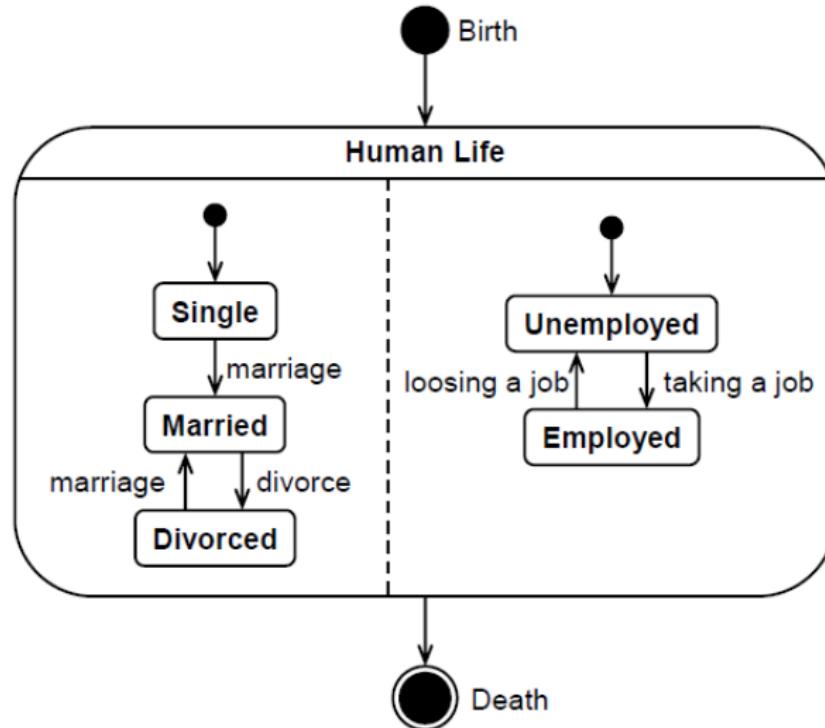
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary





Composite State

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

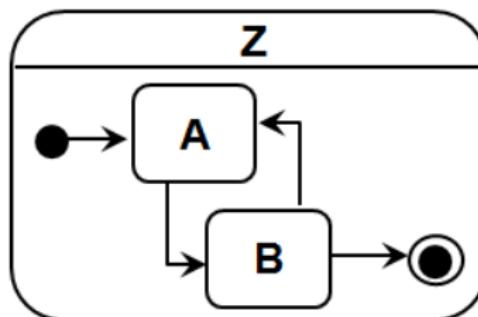
Module
Summary

disjoint sub-states (OR-

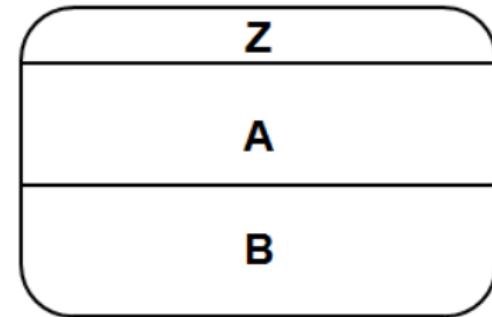
Refinement): Exactly one substate
is active when the superstate is
active

parallel sub-states (AND-

Refinement): All substates are
active when the superstate is active



Either A or B is active, when Z is active



Both A and B are active, when Z is active



Submachine State

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- An **Orthogonal submachine state** specifies the insertion of the specification of a submachine state machine
- The same state machine may be a submachine more than once in the context of a single containing state machine
- Submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Submachine State

Module 36

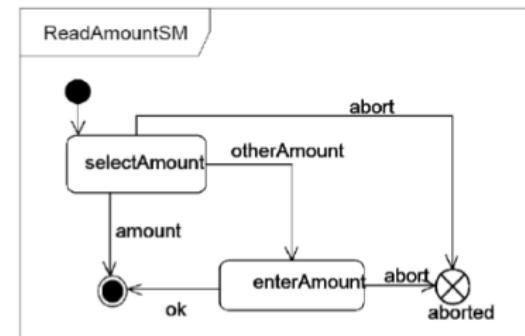
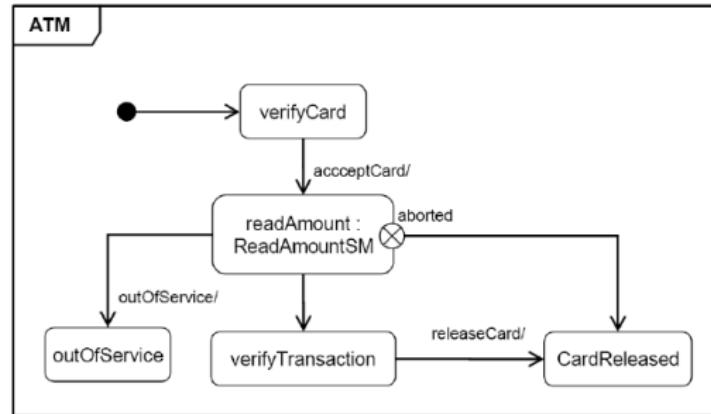
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary





Behavioral State Diagram of an ATM

Module 36

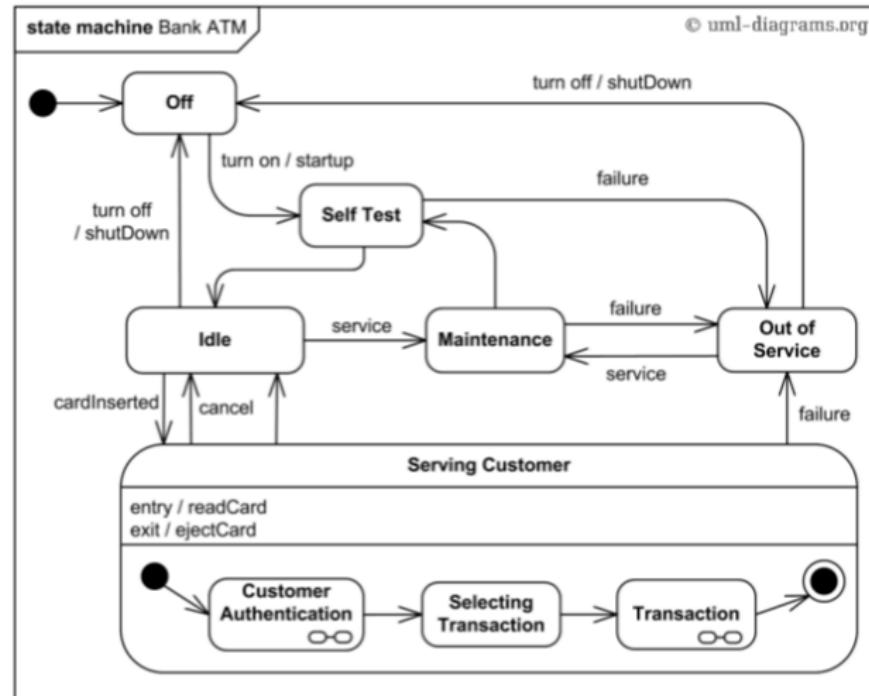
Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral State

Module
Summary



Source: <http://agilemodeling.com/style/collaborationDiagram.htm> (20-Aug-16)



Module Summary

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

- State Machine Diagrams are introduced.
- Behavioral State Diagrams are discussed. The various types like Simple, Composite and Submachine State are explained.



Instructor and TAs

Module 36

Partha Pratim
Das

Objectives &
Outline

State Machine
Diagrams

Behavioral State
Machine
Vertex
Behavioral
State

Module
Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point

Pseudostate
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

Module 37: Object Oriented Analysis & Design

State Machine Diagram: Part 2

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

● Understanding State Machine Diagrams



Module Outline

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate
Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

- What are State Machine Diagrams?
 - Behavioral State Machine
 - Vertex
 - Behavioral State
 - Pseudostate
 - Final State
 - Behavioral Transition
 - Protocol State Machine
 - State
 - Transition
- State Machine Diagram for LMS



Behavioral State Diagram of an ATM: RECAP (Module 34)

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point

Pseudostate

Choice

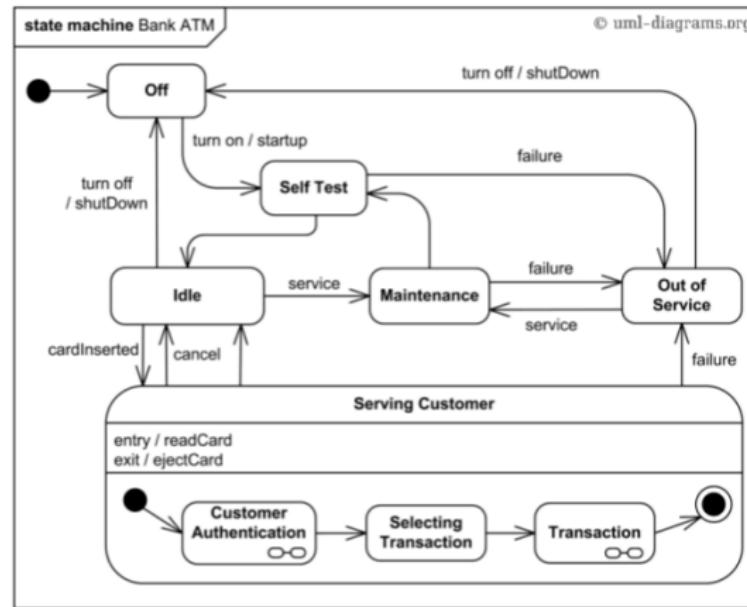
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: <http://agilemodeling.com/style/collaborationDiagram.htm> (24-Aug-16)

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das



Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

- Pseudostates (abstract vertex) are typically used to connect multiple transitions into more complex state transitions paths
- Pseudostates include
 - initial pseudostate
 - terminate pseudostate
 - entry point
 - exit point
 - choice
 - join
 - fork
 - junction
 - shallow history pseudostate
 - deep history pseudostate

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Initial and Terminal Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

Initial pseudostate: Source for a single transition to the default state of a composite state.

Notation: Small solid filled circle

Terminal pseudostate: implies termination of execution of the state.

Notation: cross



Initial pseudostate transitions to Waiting for User

Input state



Transition to terminate pseudostate

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Entry point and Exit point Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

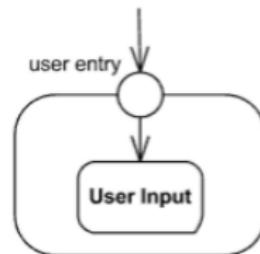
Module
Summary

Entry point pseudostate: is an entry point of a state machine or composite state.

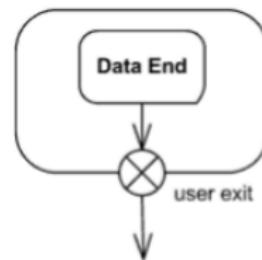
Notation: small circle on the border of the state machine diagram

Exit point pseudostate: is an exit point of a state machine or composite state.

Notation: small circle with a cross on the border of the state machine diagram



Entry point user entry



Exit point user exit

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Choice Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

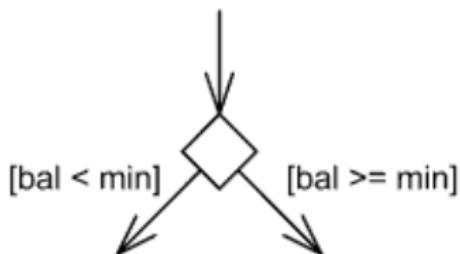
Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

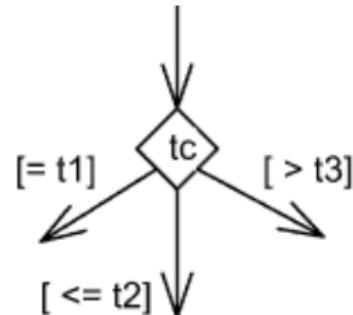
Module
Summary

Choice pseudostate : realizes a dynamic conditional branch.

Notation: diamond-shaped symbol



Select outgoing transition based on condition



Choice based on guards applied to the value inside
diamond

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Fork and Join Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

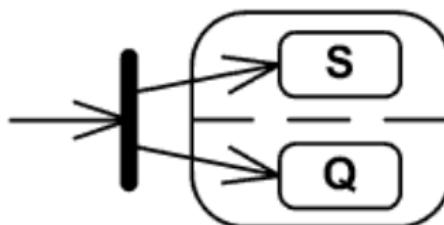
Module
Summary

Fork pseudostate : splits an incoming transition into two or more transitions terminating on target vertices in different regions .

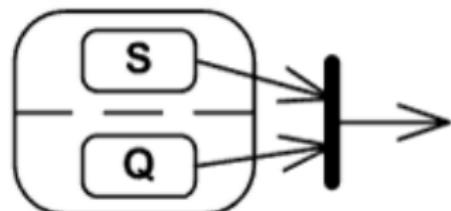
Notation: short heavy bar

Join pseudostate : merges several transitions originating from source vertices in different regions .

Notation: short heavy bar



Fork splits transition into two transitions



Join merges transitions into single transition

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Junction, Shallow history and Deep history Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

- **Junction pseudostate** vertices are vertices that are used to chain together multiple transitions.
- **Shallow history pseudostate** represents the most recent active substate of its containing state
- **Deep history pseudostate** represents the most recent active configuration of the composite state that directly contains this pseudostate

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Junction Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

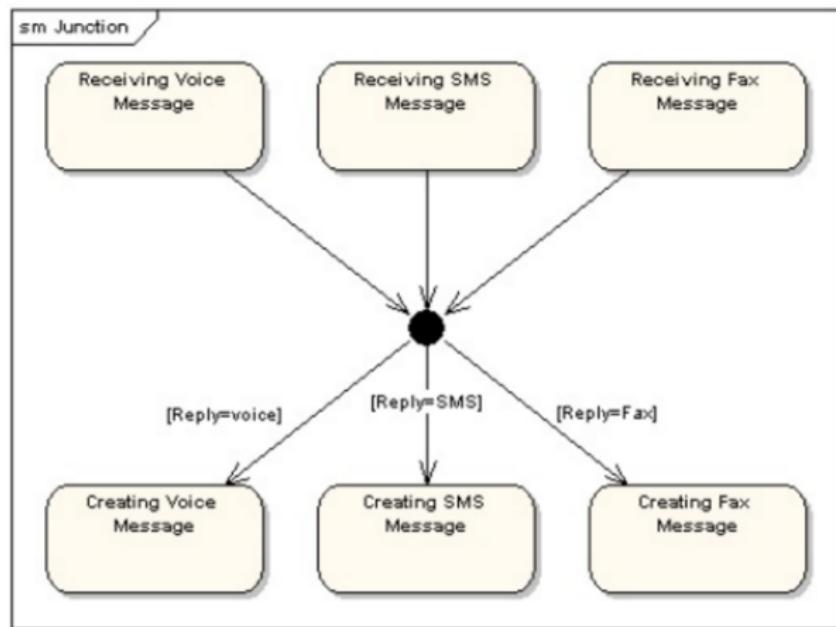
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: url: <http://www.slideshare.net/artgreen/lecture08-7433282> (24-Aug-16)

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das



Junction Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

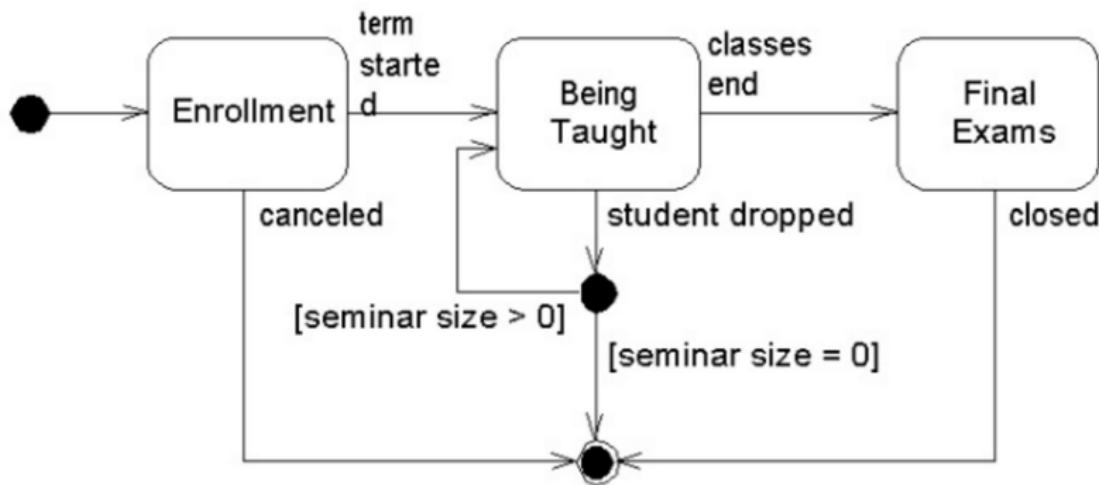
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: url: <http://www.slideshare.net/artgreen/lecture08-7433282> (24-Aug-16)



Shallow History Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

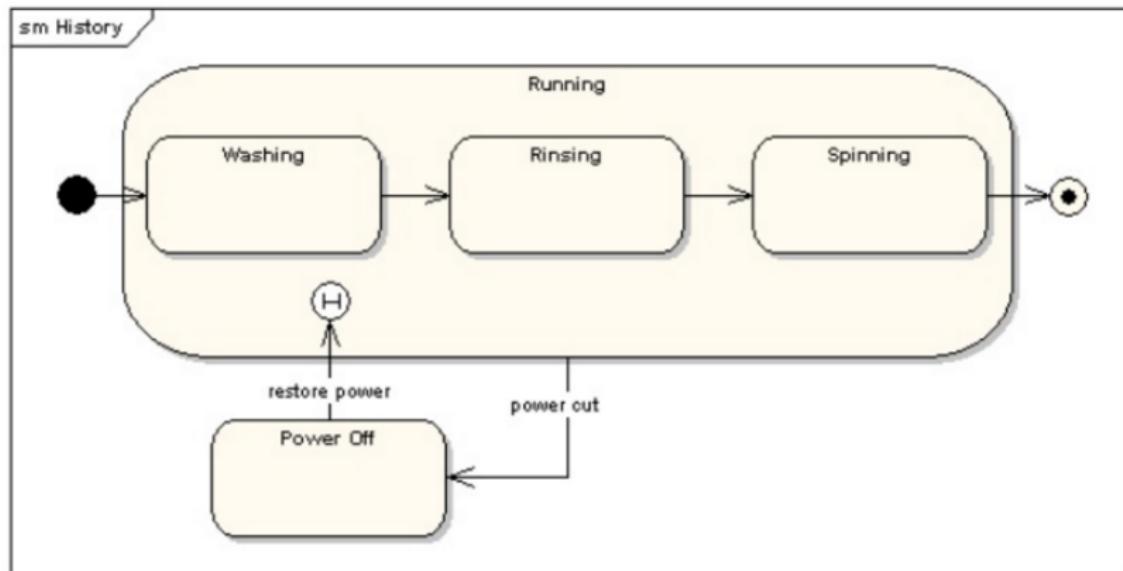
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: url: <http://www.slideshare.net/artgreen/lecture08-7433282> (24-Aug-16)

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das

13



Shallow History Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

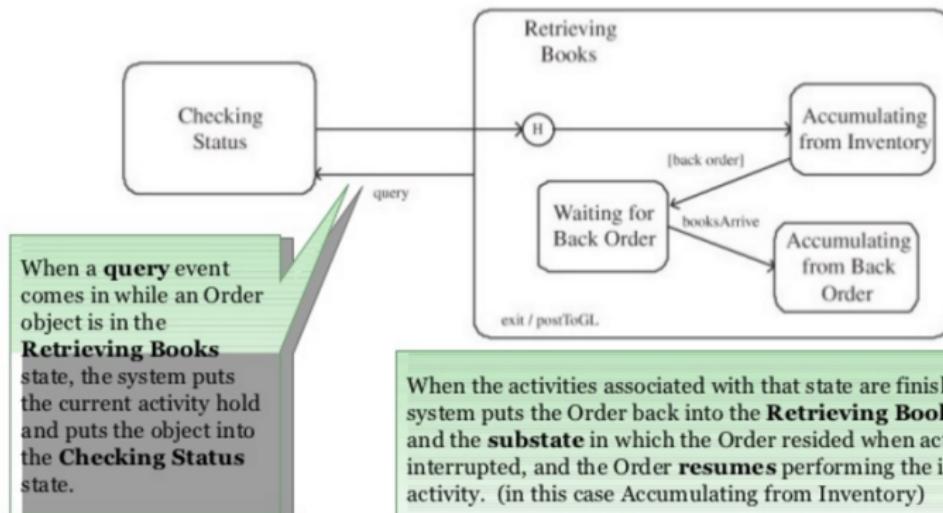
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: url: <http://www.slideshare.net/artgreen/lecture08-7433282> (24-Aug-16)



Deep History Pseudostate

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

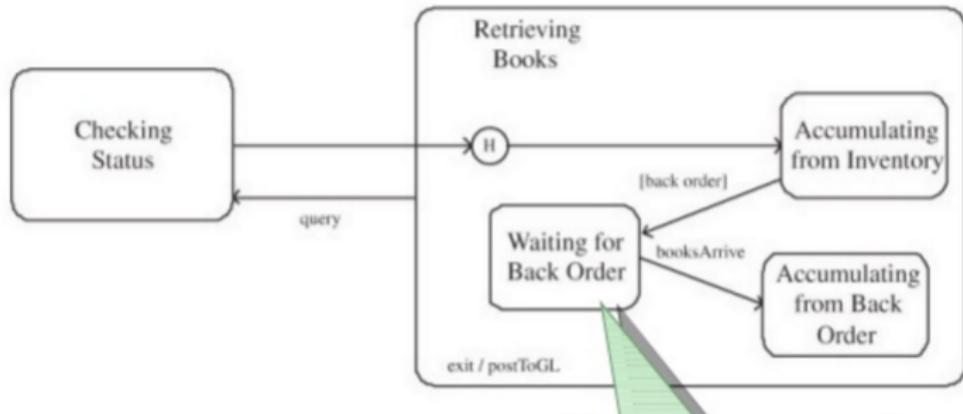
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



In this case, that execution should resume with the object in the **Waiting for Back Order** substate.



Final State

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point

Pseudostate

Choice
Pseudostate

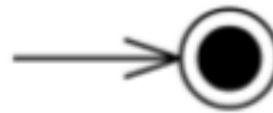
Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

- **Final state** is a special kind of state signifying that the enclosing region is completed.
- *Notation:* circle surrounding a small solid filled circle



Transition to final state

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Example

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

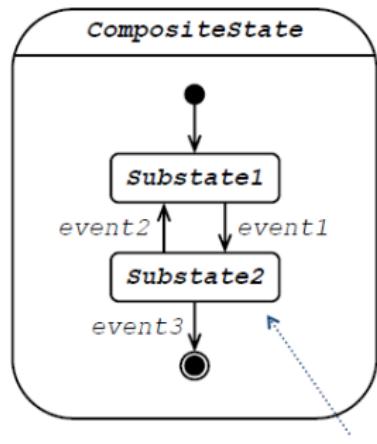
Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

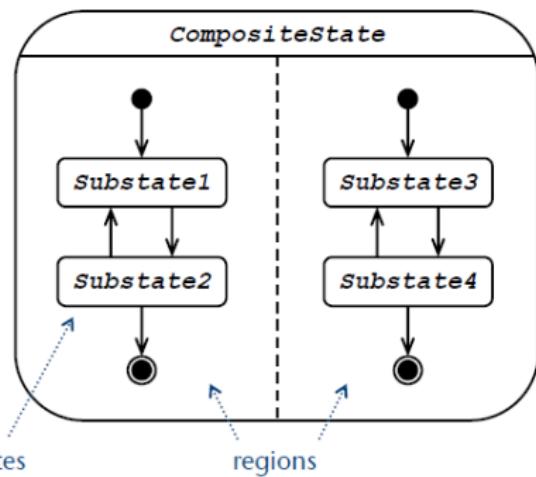
Behavioral
Transition

Module
Summary

Non-orthogonal
composite state:



Orthogonal composite state:





Example

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

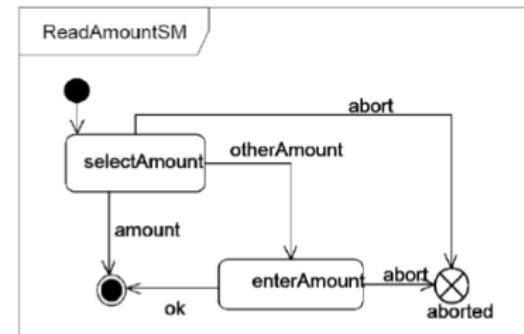
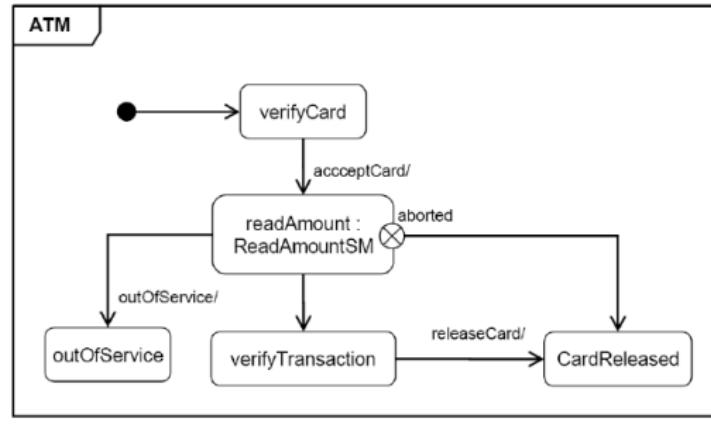
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary





Toaster_oven states

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

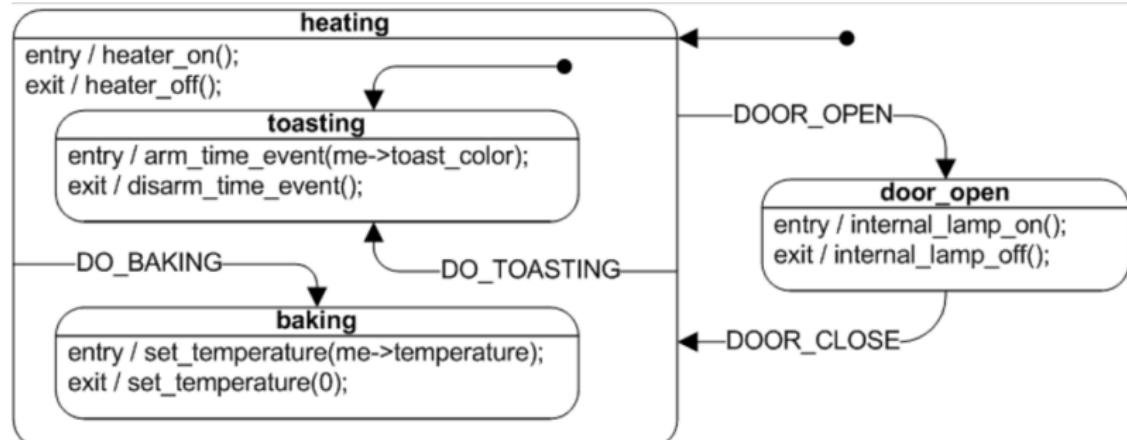
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: url: https://en.wikipedia.org/wiki/UML_state_machine#/media/File:UML_state_machine_Fig5.png
(20-Aug-16)



Calculator

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point

Pseudostate

Choice

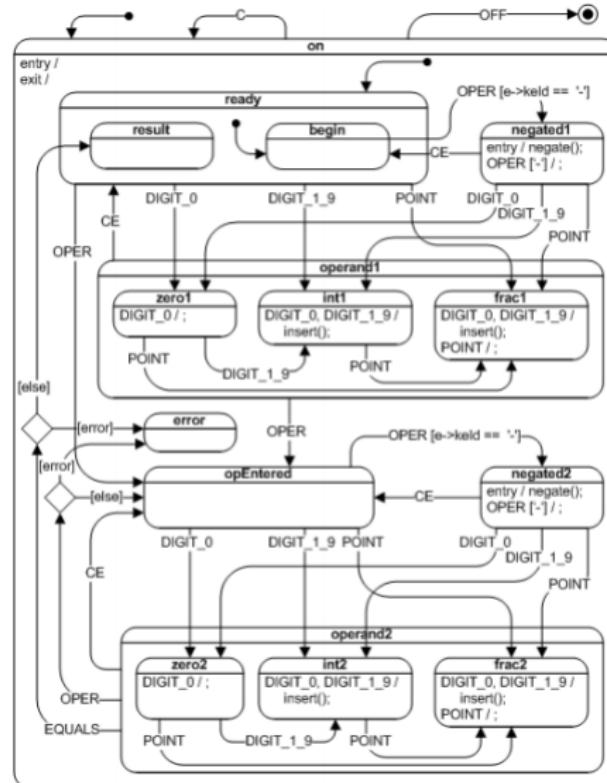
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary





Behavioral Transition

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

- A transition is a directed relationship between a source vertex and a target vertex
- The default notation for a behavioral transition are
 - `transition ::= [triggers] [guard] ['/' behavior-expression]`
 - `triggers ::= trigger [',' trigger]*`
 - `guard ::= '[' constraint ']`

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Behavioral State Diagram of an ATM

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point

Pseudostate

Choice

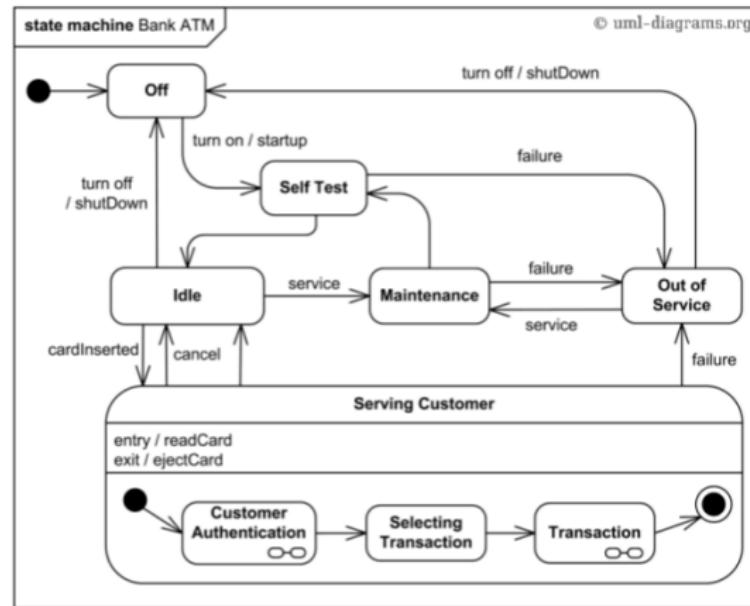
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: <http://agilemodeling.com/style/collaborationDiagram.htm> (20-Aug-16)

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das

22



OS Process

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point

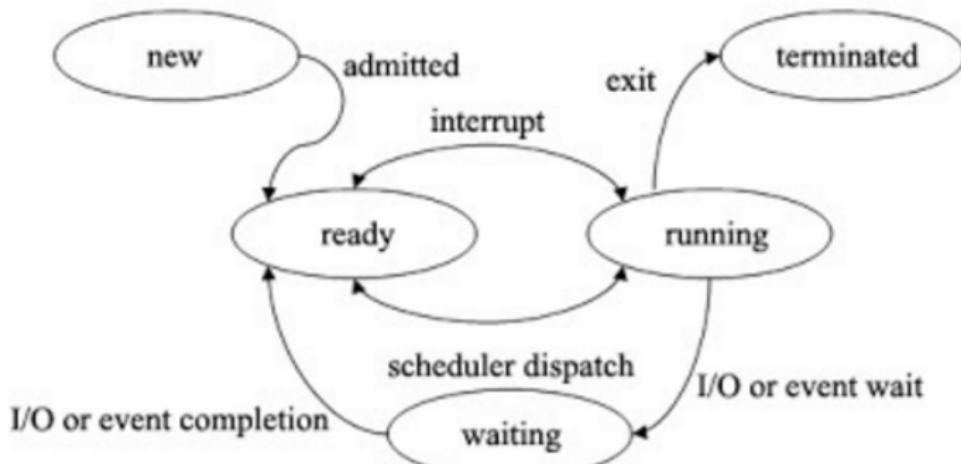
Pseudostate
Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary



Source: url <https://pwntoken.wordpress.com/2014/11/11/4-operating-system-process-state-diagram-and-cpu-scheduling-basics/>

(24-Aug-16)

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das

23



Module Summary

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate

Initial and
Terminal
Pseudostate

Entry point and
Exit point
Pseudostate

Choice
Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

- State Machine Diagrams are introduced.
- Pseudostates and Behavioral Transition of Behavioral State Diagrams are discussed.



Instructor and TAs

Module 37

Partha Pratim
Das

Objectives &
Outline

Pseudostate
Initial and
Terminal
Pseudostate

Entry point and
Exit point

Pseudostate

Choice

Pseudostate

Fork and Join
Pseudostate

Junction,
Shallow history
and Deep history
Pseudostate

Behavioral
Transition

Module
Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

Module 38: Object Oriented Analysis & Design

State Machine Diagram: Part 3

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

● Understanding State Machine Diagrams



Module Outline

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- What are State Machine Diagrams?
 - Behavioral State Machine
 - Vertex
 - Behavioral State
 - Pseudostate
 - Final State
 - Behavioral Transition
 - Protocol State Machine
 - State
 - Transition
- State Machine Diagram for LMS



Behavioral State Diagram of an ATM: RECAP (Module 34)

Module 38

Partha Pratim
Das

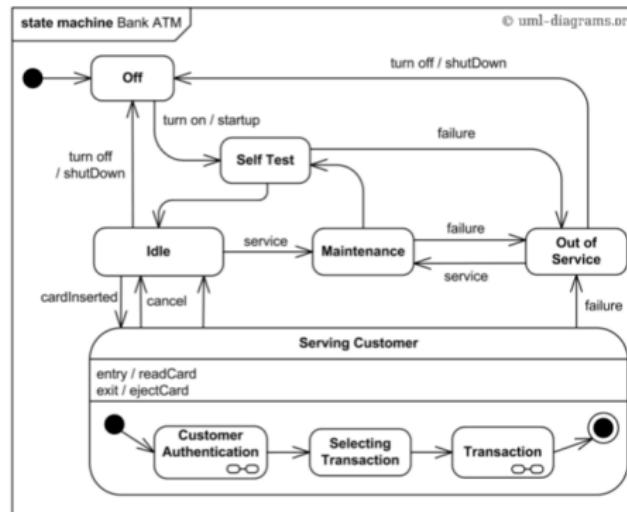
Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary



Source: <http://agilemodeling.com/style/collaborationDiagram.htm> (20-Aug-16)



Protocol State Machine

Module 38

Partha Pratim
Das

Objectives &
Outline

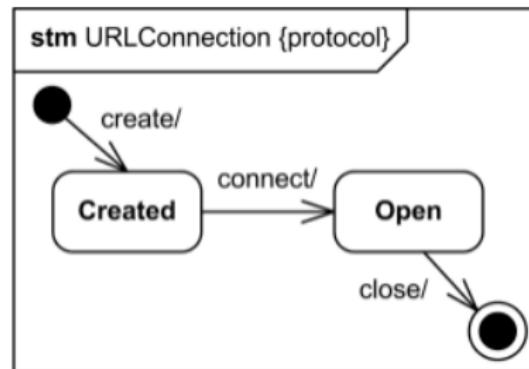
Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- Protocol state machine is a specialization of behavioral state machine and is used to express usage protocol or lifecycle of a class.
- It specifies which operations of the classifier can be called in which state and under which condition.
- It majorly consists of Protocol State and Protocol State Transitions
- The keyword {protocol} is used to distinguish protocol state diagrams



Protocol state machine for URLConnection class



Protocol State

Module 38

Partha Pratim
Das

Objectives &
Outline

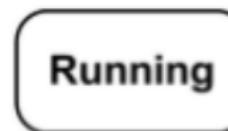
Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- The protocol states present an **external** view of the class that is exposed to its clients
- The states of protocol state machines are exposed to the users of their context classes
- **States of a protocol state machine cannot have entry, exit, or do activity actions.**



Simple protocol state Running.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Protocol State

Module 38

Partha Pratim
Das

Objectives &
Outline

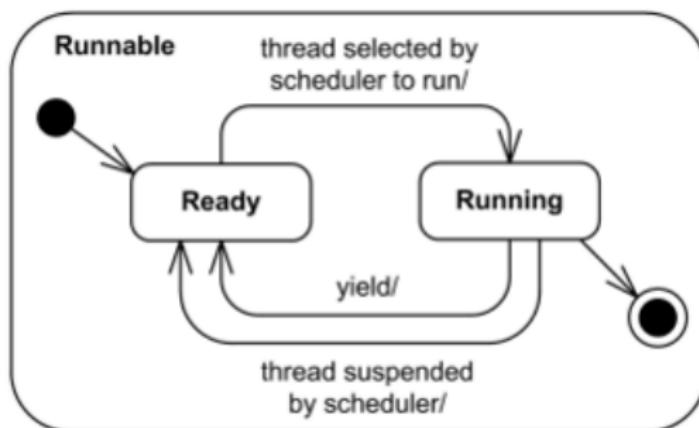
Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- Protocol state machines can have **submachine states, composite states**, and concurrent regions.
- Concurrent regions make it possible to express protocol where an instance can have several active states simultaneously



Simple composite protocol state Runnable

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Keyboard Operation

Module 38

Partha Pratim
Das

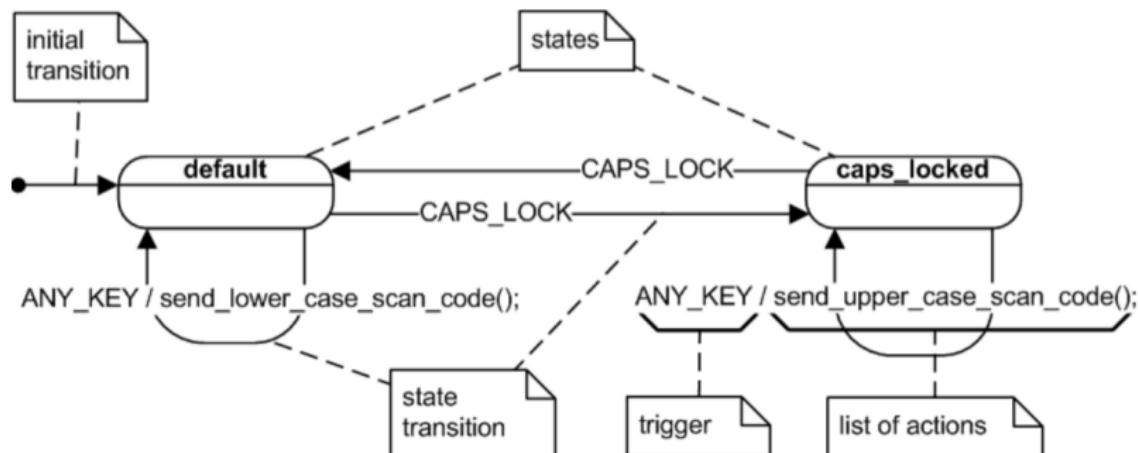
Objectives &
Outline

Protocol State
Machine

Protocol State
Transition

State Machine
Diagram for
LMS

Module
Summary



Source: url:

https://en.wikipedia.org/wiki/UML_state_machine#/media/File:UML_state_machine_Fig1.png(24-Aug-16)



An Elevator: RECAP (MODULE 11)

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

Elevator	
- <i>make</i> :	String
- <i>model</i> :	String
- <i>max_Persons</i> :	Integer
- <i>max_Load</i> :	Integer
- <i>floor</i> :	{1, 2, X}
- <i>isMoving</i> :	{Still, Up, Down}
- <i>doorOpen</i> :	Bool
+ Up()	// Go Up
+ Down()	// Go Down
+ Floor()	// Current Floor
+ Moving()	// IsMoving?
+ Stop()	
+ Open()	// Open Door
+ Close()	// Close Door

- **Static** Properties – *does not change* for an instance
- **Dynamic** Values – *changes regularly* for an instance



An Elevator – States: RECAP (MODULE 11)

Module 38

Partha Pratim
Das

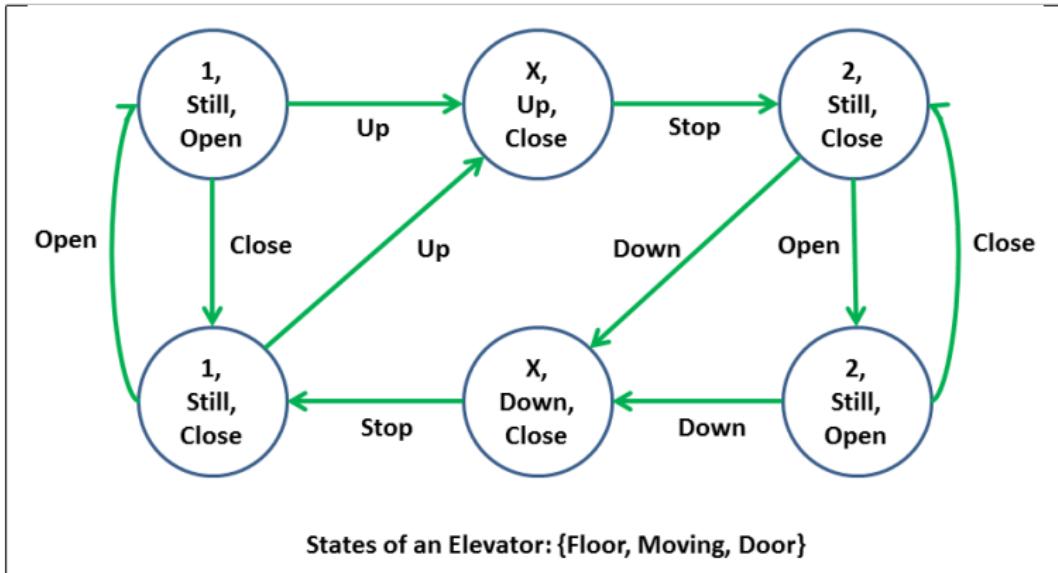
Objectives &
Outline

Protocol State
Machine

Protocol State
Transition

State Machine
Diagram for
LMS

Module
Summary



The elevator moves through these states as it operates



Protocol Transition

Module 38

Partha Pratim
Das

Objectives &
Outline

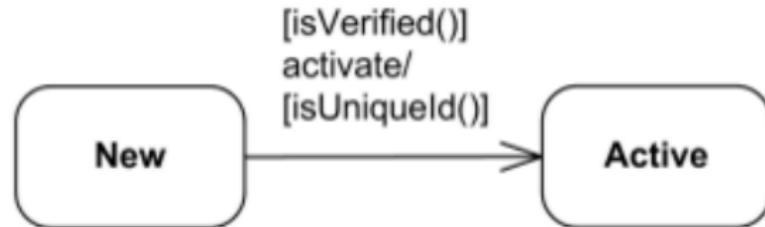
Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- A protocol transition is **specialization of (behavioral) transition** used for the protocol state machines which specifies a legal transition for an operation
- Protocol transition has the following **features**: a pre-condition (guard), trigger, and a post-condition
- Compound transitions can be used for protocol state machines.
- **Notation:** *Transition arrow from the source vertex to the target vertex, with optional text describing transition*



Protocol transition from New to the Active state

with pre-condition (guard), trigger, and a post-condition.

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Protocol State Machine

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

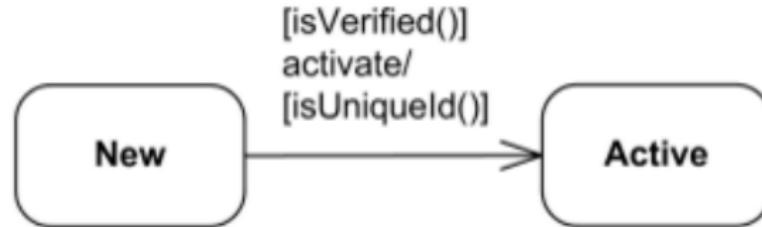
Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- The textual notation for a protocol transition:

- $\text{protocol-transition} ::= [\text{ pre-condition }] \text{ trigger } '/' [\text{ post-condition }]$
- $\text{pre-condition} ::= '[' \text{ constraint } '']$
- $\text{post-condition} ::= '[' \text{ constraint } '']$



Protocol transition from New to the Active state

with pre-condition (guard), trigger, and a post-condition.

Source: *UML 2.5 Diagrams Overview*: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Online Shopping User Account

Module 38

Partha Pratim
Das

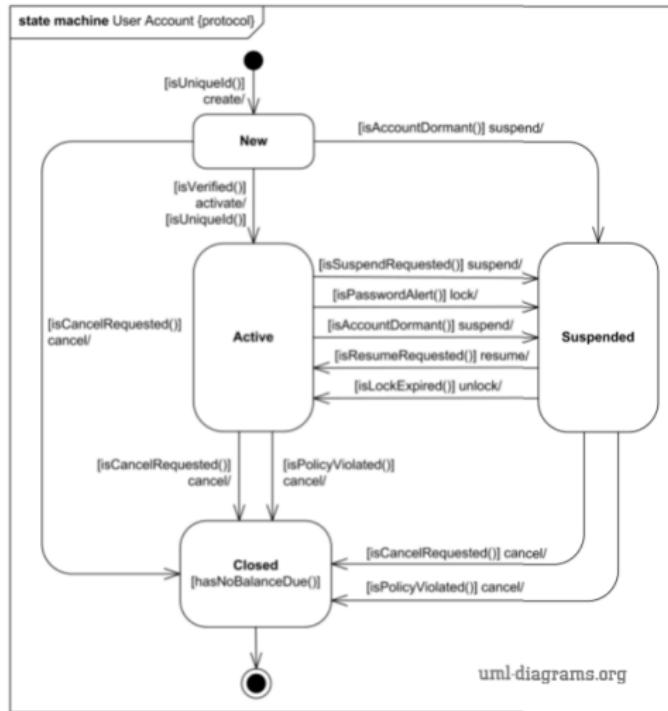
Objectives &
Outline

Protocol State
Machine

Protocol State
Transition

State Machine
Diagram for
LMS

Module
Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)

NPTEL MOOCs Object Oriented Analysis and Design

Partha Pratim Das

13



Online Shopping User Account – Annotated

Module 38

Partha Pratim
Das

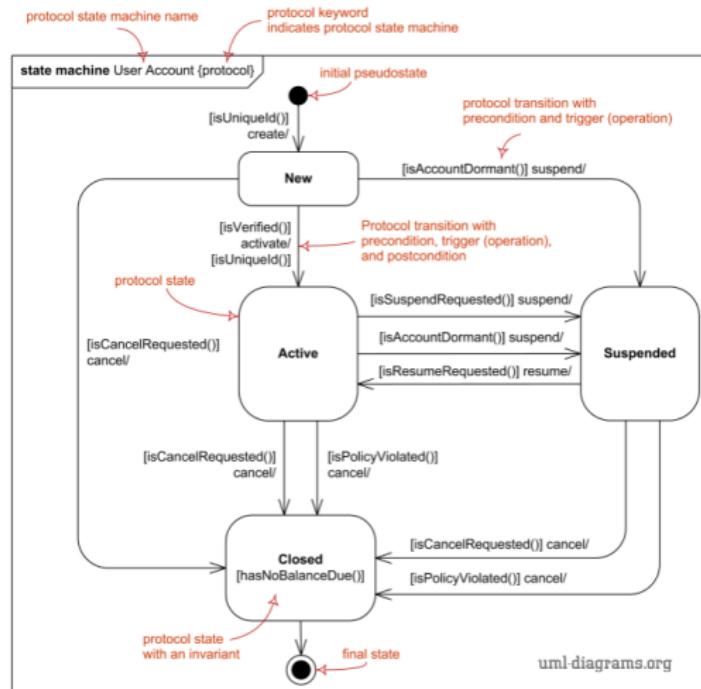
Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



State Machine Diagram for LMS

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

We will now derive the State Machine Diagram for LMS. The steps are:

- Identify the states
- Identify the transitions (including pre-condition, post-condition and triggers) which changes one state into the other state
- Derive the final State Machine Diagram



States of LMS

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

We identify the states of the Leave Life-cycle

- New
- Requested
- Approved
- Regretted
- Availed
- Canceled
- Archived



Transitions of LMS

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

We identify the states of the Leave Life-cycle

- New to Requested: Req[valid] / Block Dates
- Requested to Approved: App / Notify
- Requested to Regretted: Req / Notify Release Dates
- Approved to Canceled: Can / Notify Release Dates
- Approved to Availed: Avl/ Deduct Leave
- Approved to Regretted: Req / Notify Release Dates
- Regretted to Archived: batch
- Regretted to Approved: Req / Notify Block Dates
- Availed to Archived: batch
- Canceled to Archived: batch
- Canceled to Requested: Req / Notify Block Dates



State Machine Diagram of LMS

Module 38

Partha Pratim
Das

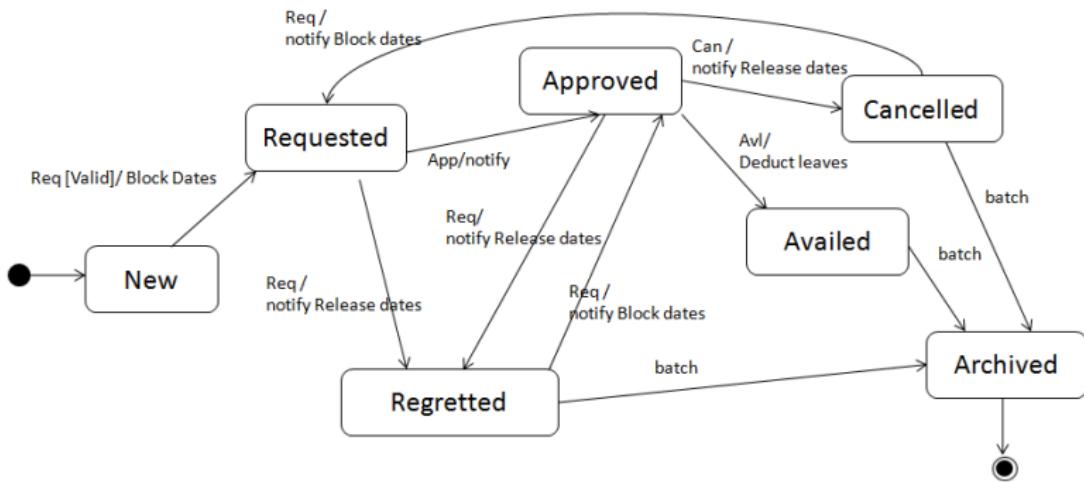
Objectives &
Outline

Protocol State
Machine

Protocol State
Transition

State Machine
Diagram for
LMS

Module
Summary





Module Summary

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

- Protocol State Diagram is introduced
- The states and transitions of the Protocol State Diagram is discussed
- The State Machine diagram fro LMS is derived



Instructor and TAs

Module 38

Partha Pratim
Das

Objectives &
Outline

Protocol State
Machine

Protocol State
Protocol
Transition

State Machine
Diagram for
LMS

Module
Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

Module 39: Object Oriented Analysis & Design

Various UML Diagrams

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams
Timing Diagram

Structural
Diagrams
Component
Diagram
Deployment
Diagram
Composite
Structure
Diagram
Package
Diagram

Summary

- Familiarization with Component, Deployment, Composite Structure, Package, and Timing Diagrams
- Completion of UML Diagrams



Module Outline

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams
Timing Diagram

Structural
Diagrams

Component
Diagram
Deployment
Diagram
Composite
Structure
Diagram
Package
Diagram

Summary

- Behavioral Diagrams
 - Timing Diagram
- Structural Diagrams
 - Component Diagram
 - Deployment Diagram
 - Composite Structure Diagram
 - Package Diagram



Diagram Classification: UML 2.5

RECAP (Module 21)

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

- **Structural Diagrams** show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other
 - The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts
- **Behavioral Diagrams** show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (10-Aug-16)



Diagram Classification: UML 2.5 RECAP (Module 21)

Module 39

Partha Pratim
Das

Objectives &
Outline

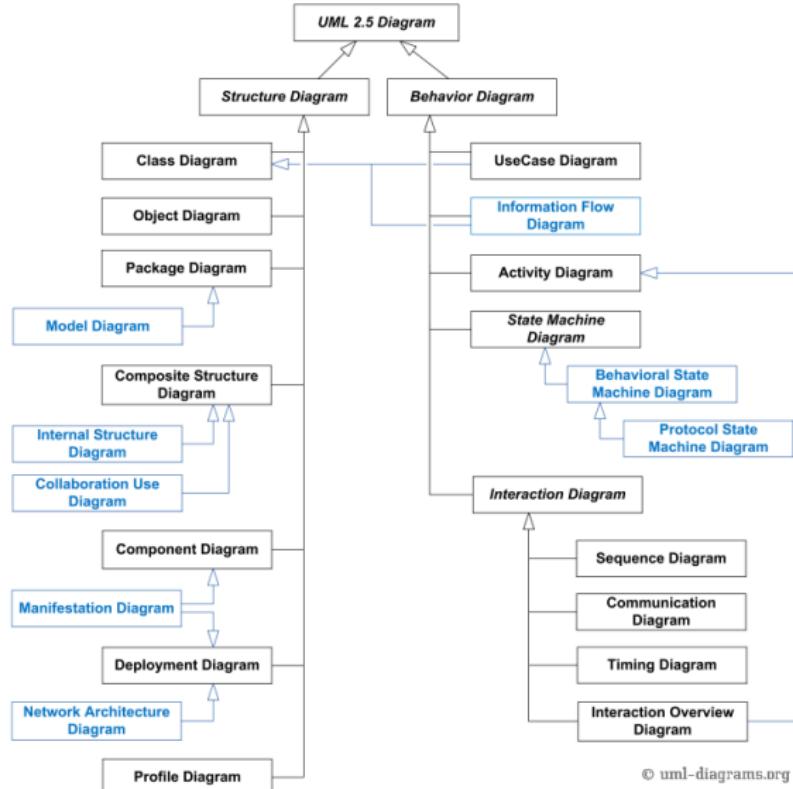
Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram
Deployment
Diagram
Composite
Structure
Diagram
Package
Diagram

Summary



© uml-diagrams.org



Timing Diagrams

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

- Timing diagram is a **behavioral diagram** which focuses on conditions changing within and among lifelines along a linear time axis
- Timing Diagrams are built during **Analysis** and **Design** phases of SDLC
- Both individual class and interactions of classes are described, focusing attention on time of events causing changes in the modeled conditions of the lifelines.
- The major components of a Timing Diagram are:
 - Lifeline
 - State or Condition Timeline
 - Duration constraint
 - Time constraint
 - Destruction Occurrence



Timing Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

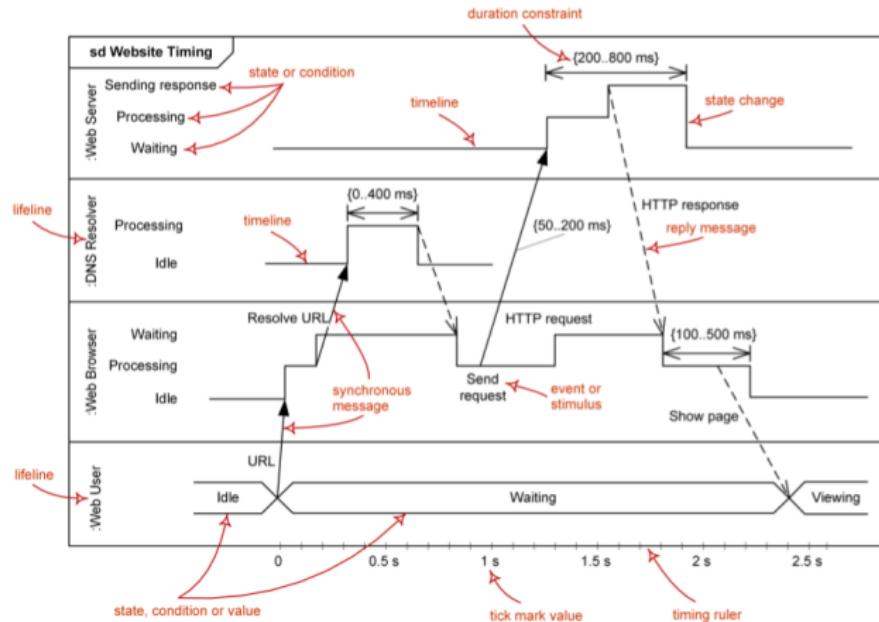
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Component Diagram

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

- Component diagram is a **structure diagram** which shows **components**, **required interfaces**, **ports**, and **relationships** between them
- Component diagrams are built during **Design** and **Implementation** phases of SDLC
- This type of diagrams is used for **Component-Based Development (CBD)**, to describe systems with **Service-Oriented Architecture (SOA)**
- Components in UML could represent **logical components** (business components, process components) and **physical components** (CORBA components, EJB components, COM+ and .NET components, WSDL components)
- The major components of a Component Diagram are:
 - Component
 - Interface
 - Provided interface
 - Required interface
 - Class
 - Artifact
 - Port, Connector



Component Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

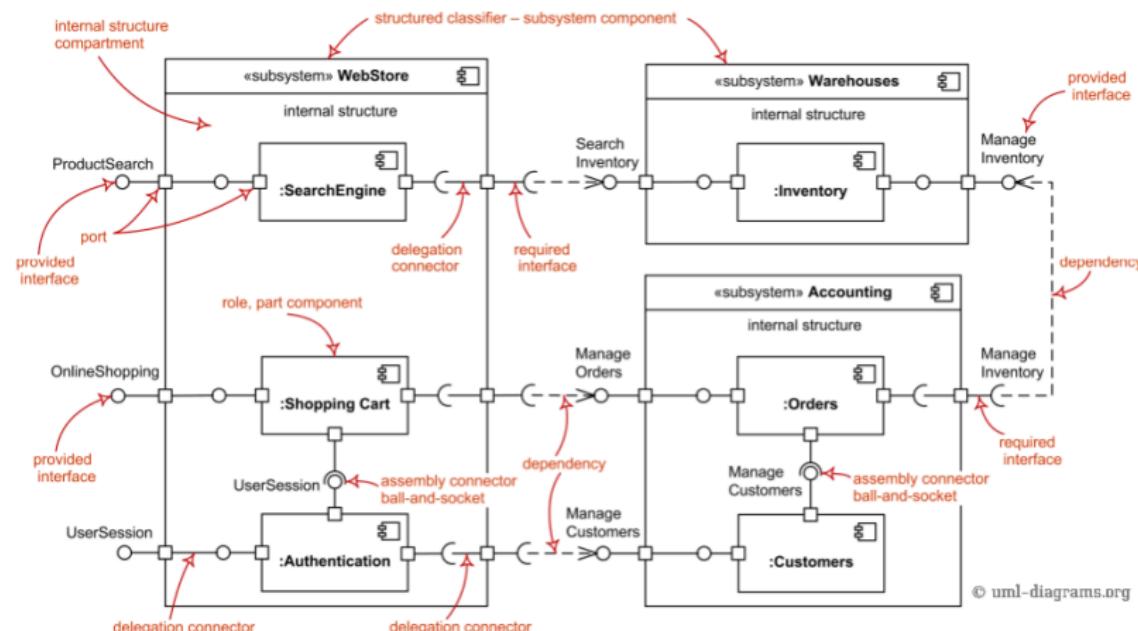
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



© uml-diagrams.org

Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Deployment Diagram

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

- Deployment diagram is a **structure diagram** which shows architecture of the system as **deployment (distribution)** of software artifacts to deployment targets.
- Deployment Diagrams are built during **Design** and **Implementation** phases of SDLC
- **Artifacts** represent concrete elements in the physical world that are the result of a development process
- Examples of artifacts are executable files, libraries, archives, database schemas, configuration files
- Deployment target is usually represented by a **node** which is either hardware device or some software execution environment
- The major components of a Deployment Diagram are:
 - Artifacts (instances)
 - Deployment Targets (instances)



Deployment Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

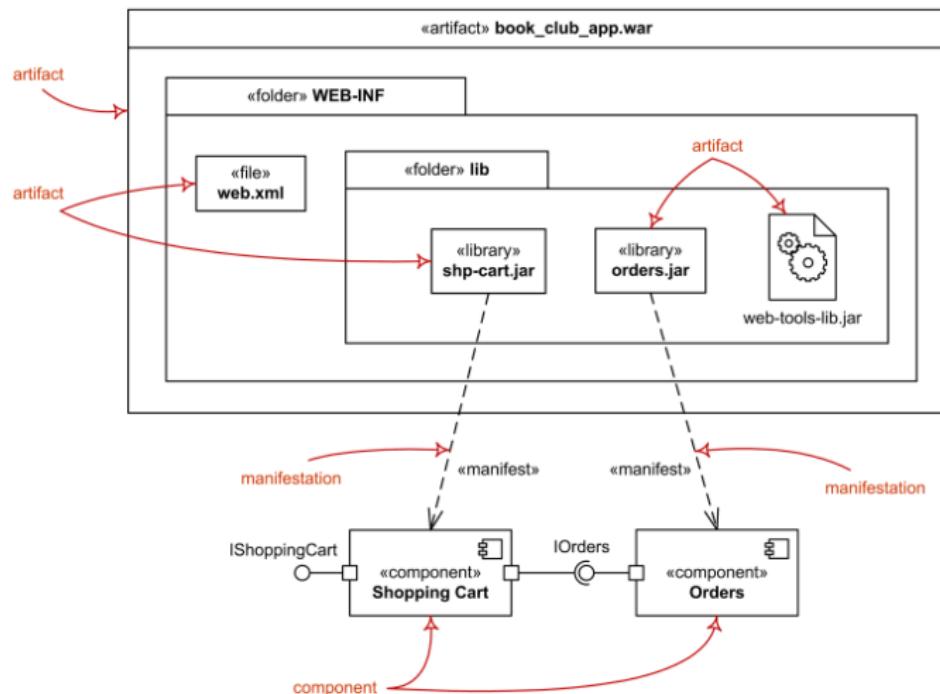
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Specification Level Deployment Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

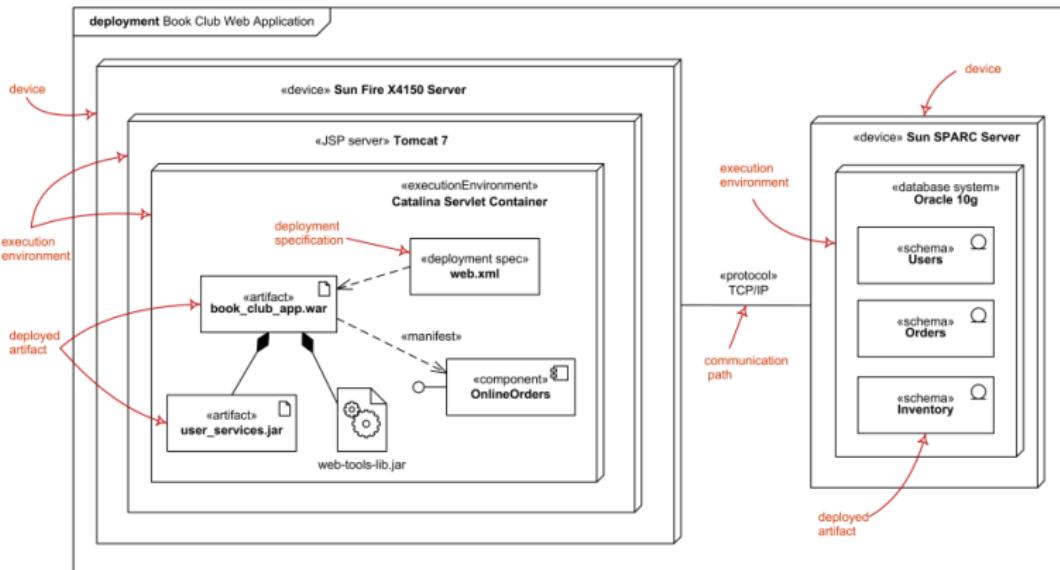
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Instance Level Deployment Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

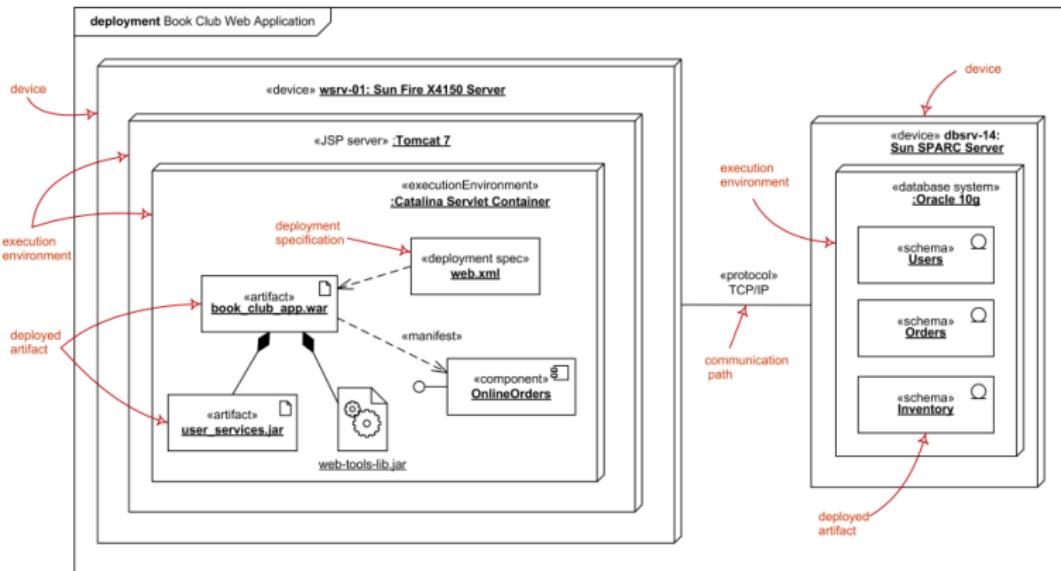
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Composite Structure Diagram

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams
Timing Diagram

Structural
Diagrams
Component
Diagram
Deployment
Diagram
Composite
Structure
Diagram

Package
Diagram

Summary

- Composite Structure Diagram is a **structure diagram** which is used to show
 - Internal structure of a classifier - **internal structure diagram**
 - A behavior of a collaboration - **collaboration use diagram**
 - Class interactions with environment through **ports**
- **Internal structure diagram** shows internal structure of a classifier - a decomposition of that classifier into its properties, parts and relationships
- **Collaboration use** represents one particular use (occurrence) or application of the pattern described by a collaboration to a specific situation involving specific classes or instances playing the roles of the collaboration
- Composite Structure Diagrams are built during **Design** and **Implementation** phases of SDLC



Internal Structure Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

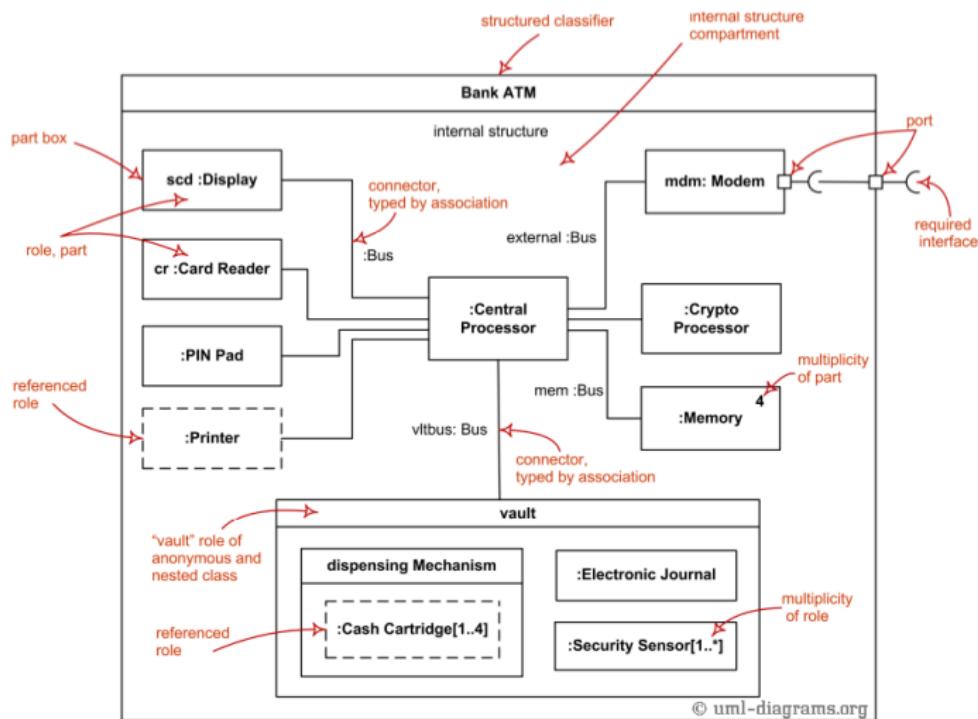
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Collaboration Use Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

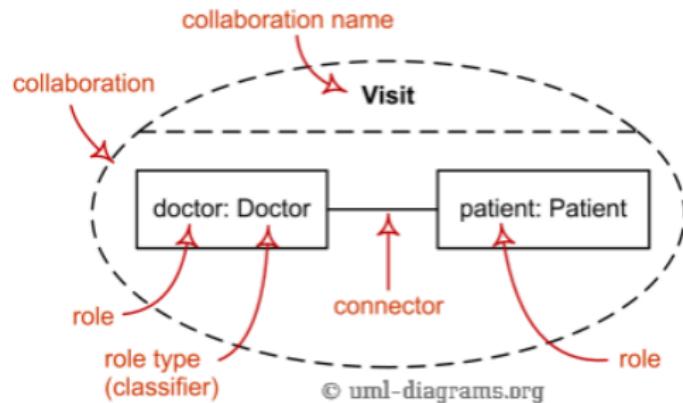
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Collaboration Use Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

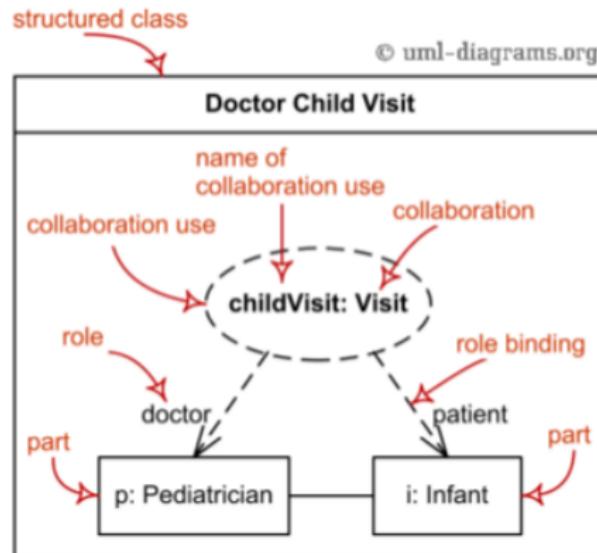
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Package Diagram

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

- Package diagram is a **structure diagram** which shows structure of the designed system at the level of packages
- A system at a higher level can be grouped into logical sections, consisting of group of related diagrams
- Package Diagrams are built during **Design** and **Implementation** phases of SDLC
- A package has similar properties like classes, for example, visibility and associations among classes, but with the difference, that the property assigned to package will apply to all its component classes
- The major components of a Package Diagram are:
 - Package
 - Dependency
 - Import
 - Access
 - Merge



Package Diagram – Annotated

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

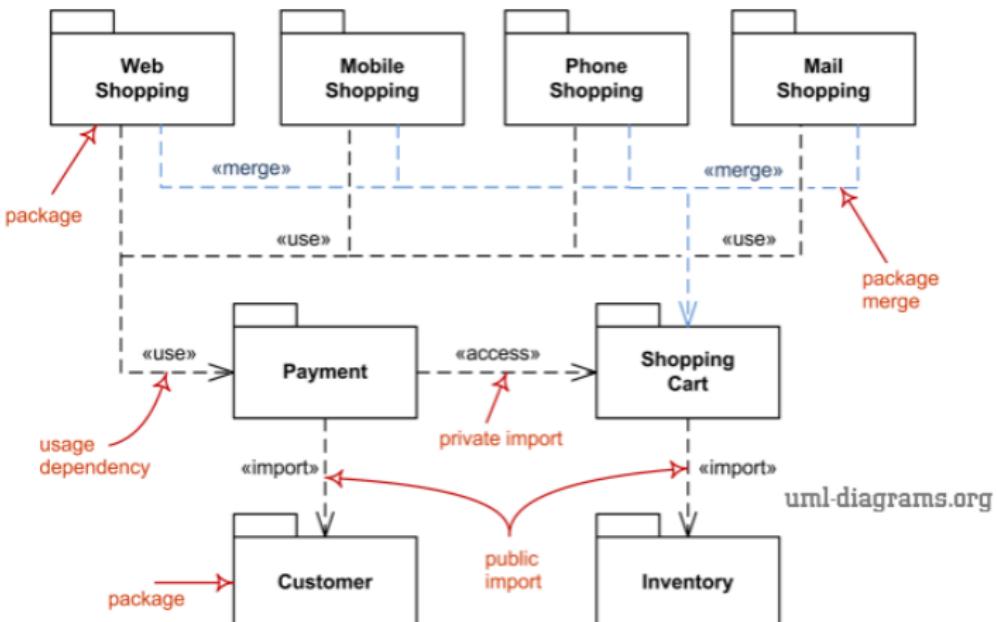
Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary



Source: UML 2.5 Diagrams Overview: <http://www.uml-diagrams.org/uml-25-diagrams.html> (24-Aug-16)



Module Summary

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

- Familiarized with Component, Deployment, Composite Structure, Package, and Timing Diagrams
- Annotated Examples are illustrated



Instructor and TAs

Module 39

Partha Pratim
Das

Objectives &
Outline

Behavioral
Diagrams

Timing Diagram

Structural
Diagrams

Component
Diagram

Deployment
Diagram

Composite
Structure
Diagram

Package
Diagram

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655



Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

Module 40: Object Oriented Analysis & Design

Closing Comments

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick

Srijoni Majumdar

Himadri B G S Bhuyan

This presentation uses diagrams, examples and selected texts from Object-Oriented Analysis and Design – With Applications by Grady Booch et. al. (3rd Ed, 2007) with kind permission of the author



Module Objectives

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- Review OOAD Course
- Information for Examination
- What next?



Module Outline

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- Course Summary – week-wise
- Key Take-back
- Prepare for Examination
- Road Forward



Week 1: Software Complexity

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **Is Software Engineering? Why Software Projects Fail?**
- **Four Elements of Complexity:** Complexity of the Problem Domain, Difficulty of Managing the Development Process, Flexibility Possible through Software, and Problems of Characterizing the Behavior of Discrete Systems
- **Structure of Complex Systems**
- **Attributes of a Complex System:** Hierarchic Structure, Relative Primitives, Separation of Concerns, Common Patterns, and Stable Intermediate Forms
- **Organized Complexity:** Decomposition Hierarchy, Abstraction Hierarchy, Class and Object Structure, and Canonical Form
- **Disorganized Complexity:** Limited Human Capacity
- **Bringing Order to Chaos:** Role of Decomposition, Abstraction, and Hierarchy
- **Designs and Models**



Week 2: Computation Paradigms & Object Models

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **Computation:** Programming and Hardware Architecture
- **Generations of Programming Languages with typical Topology**
- **Foundations of Object Models:** Interrelationships of OOA, OOD, and OOP
- **Elements of Object Model**
 - Major Elements: Abstraction, Encapsulation, Modularity, and Hierarchy
 - Minor Elements: Typing, Concurrency, and Persistence



Week 3: Classes and Objects: Broader Perspectives

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1

Week2

Week3

Week4

Week5

Week6

Week7

Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **Nature of an Object:** State, Behavior and Identity
- **Relationships among Objects:** Links and Aggregation
- **Nature of a Class:** Interface and Implementation
- **Relationships among Classes:** Association, Inheritance, Aggregation, and Dependencies
- **Measuring the Quality of an Abstraction: Choosing Operations, Relationships, and Implementations**



Week 4: Classes and Objects: How to Identify?

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1

Week2

Week3

Week4

Week5

Week6

Week7

Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **What is Classification**

- **Extraction of Classes**

- Classification: Structural Clustering, Conceptual Clustering, and Prototyping
- Identification: Key Abstractions

- **Classes, Objects, and Relationships in LMS: Linguistic Approach and Analysis**



Week 5: Modeling with UML

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **Overview of UML**
- **UML Diagrams: Structural Diagrams and Behavioral Diagrams**
- **SDLC phases: Requirements Specification Phase, Analysis Phase, Design Phase, and Implementation Phases – Relating UML diagrams to phases**
- **Use Case Diagram**



Week 6: Modeling with UML

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **Class Diagram**
- **Sequence Diagram**



Week 7: Modeling with UML

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **Communication Diagram**
- **Activity Diagram**
- **Interaction Overview Diagram**



Week 8: Modeling with UML

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- **State Machine Diagram**
- **Timing Diagram**
- **Component Diagram**
- **Deployment Diagram**
- **Composite Structure Diagram**
- **Package Diagram Diagram**



What have we learnt?

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- Software is Complex because Systems are Complex
- OOAD helps understanding and working with Complex Systems because
 - OOAD is natural and scalable
 - OOAD is process, people and platform agnostic
- UML is a powerful modeling instrument that
 - provides a language of expression that is process, people and platform agnostic
 - aligns perfectly with object-oriented paradigm
 - works at multiple levels of details with multiple facets
- Object-Orientation is not in the tool, language or platform – it is how one looks at a system – it is in the Analysis and the Design!

Object-Orientation is in the mind of the Developer



Prepare for Examination

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1

Week2

Week3

Week4

Week5

Week6

Week7

Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- Watch the Videos
- Revise the Assignments and Solutions
- Practice Used / Worked Out Examples
 - **Leave Management System (LMS)**
 - **(Students') Assignment Management System (AMS)**
- Practice with Additional Examples
 - **(Indian) Postal Management System (PMS):**
 - **(Newspaper) Story Management System (SMS)**
 - **(Rental) Car Management Systems (CMS)**
- Study Books
 - **Object-Oriented Analysis and Design – With Applications** by *Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. Houston*, 3rd Ed., 2007
 - **Learning UML 2.0 – A Pragmatic Introduction to UML** by *Russ Miles, Kim Hamilton*, 2006. Publisher: O'Reilly Media



Road Forward

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- Practice modeling with UML
- Learn **Object Oriented Programming** in C++ or Java or Python
- Analyze, Design and Implement of moderate sized systems
- Study **Software Engineering** to understand and use processes better



Module Summary

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

- Course on OOAD concluded



Instructor and TAs

Module 40

Partha Pratim
Das

Objectives &
Outline

Course
Summary

Week1
Week2
Week3
Week4
Week5
Week6
Week7
Week8

Key Take-back

Prepare for
Examination

Road Forward

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655