Object Modeling

Object Oriented Programming
2016375 - 5

Camilo López

Outline

- The Goal of Object Modeling
- Software Development Process, an overview
- Requirements Workflow
- Analysis Workflow

The Goal of Object Modeling

The "Big Picture"

- Our goal in object modeling is to render a precise, concise, understandable object-oriented model, or "blueprint," of the system to be automated.
  This model will serve as an important tool to communicate:
  - **To the future users of the system** that we are about to build, our understanding of the system requirements.
  - **To the software development team**, the structure and function of the software that needs to be built in order to satisfy those requirements.
  - Also, it's as a schematic diagram to help the myriad folks **responsible for supporting and maintaining an application** understand its structure and function.

The Goal of Object Modeling

Modeling Methodology = Process + Notation + Tool

- According to Webster's dictionary, a **methodology** is

  A set of systematic procedures used by a discipline
  [to achieve a particular desired outcome].

- A modeling methodology, OO or otherwise, ideally involves three components:
  - **A process**: The "how to" steps for gathering the requirements and determining the abstraction to be modeled
  - **A notation**: A graphical "language" for communicating the model
  - **A tool**: An automated way of rendering the notation, typically in drag-and-drop fashion

The Goal of Object Modeling

Modeling Methodology = Process + Notation + Tool

- According to Webster's dictionary, a **methodology** is

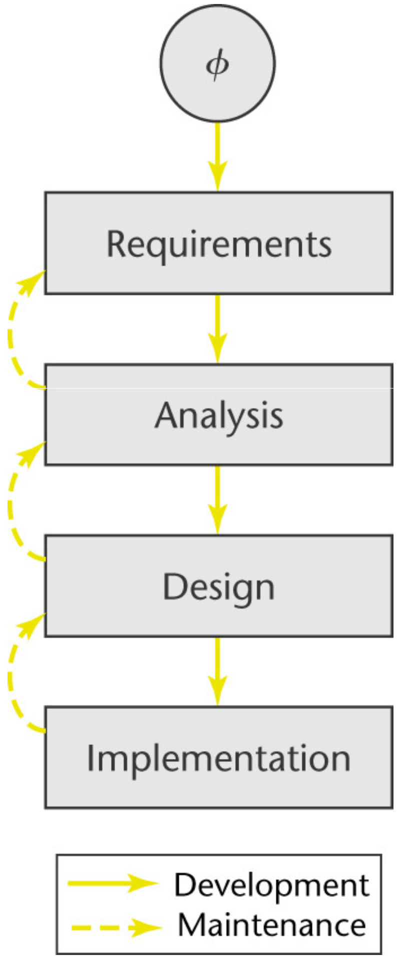
  A set of systematic procedures used by a discipline
  [to achieve a particular desired outcome].

- A modeling methodology, OO or otherwise, ideally involves three components:

PROCESS + NOTATION + TOOL

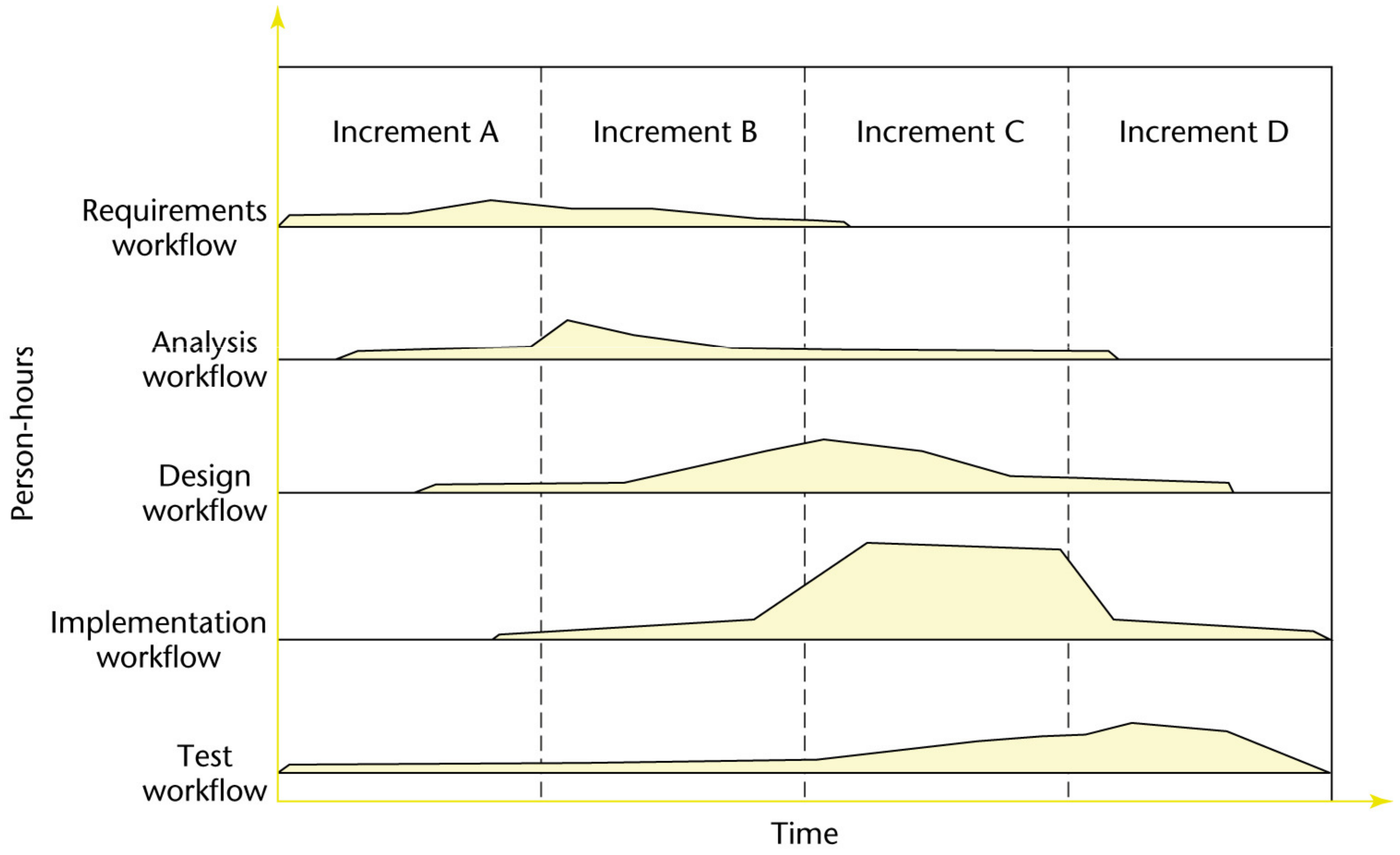
Software Development Process

The Waterfall Model



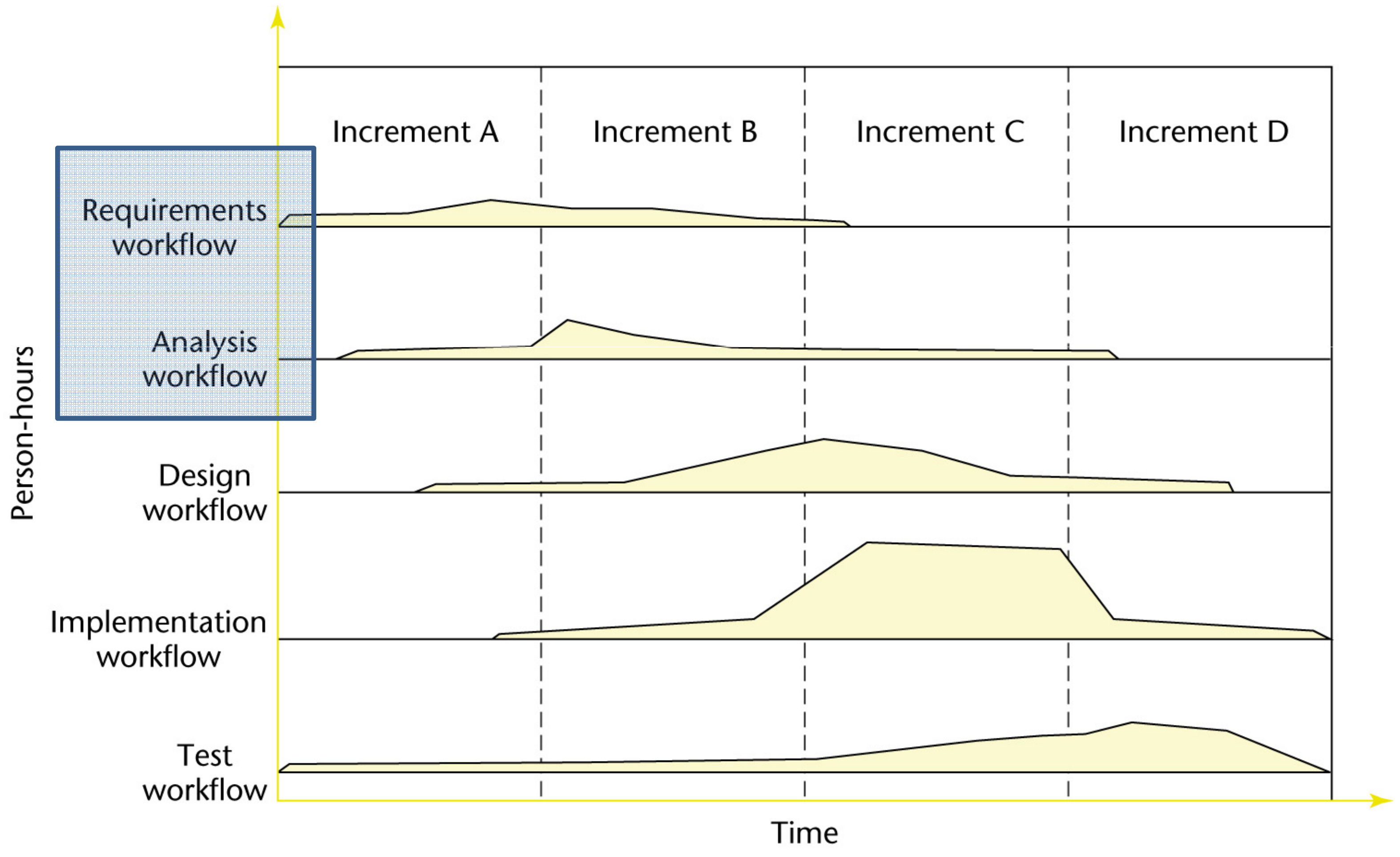
Software Development Process

Iterative and Incremental Life-Cycle Model



Software Development Process

Iterative and Incremental Life-Cycle Model



Requirements Workflow

- Functional vs. Technical (Non-functional) Requirements.
- To determine the client's needs.
  - Understanding of the application domain
  - The business model: A description of the business processes of an organization.
  - Use Case analysis.
    Use Case models an interaction between the software product itself and the users of that software product (actors).
    - An actor for every different role
    - Specify Use Cases
    - Matching Up Use Cases with Actors

Use Case Diagram:
Shows the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.

Requirements Workflow

Use Case: Specification

- Use Case name
- Brief Description
- Step by Step Description

- Pre-condition
- Post-condition
- Trigger

Requirements Workflow

Use Case: UML Diagrams

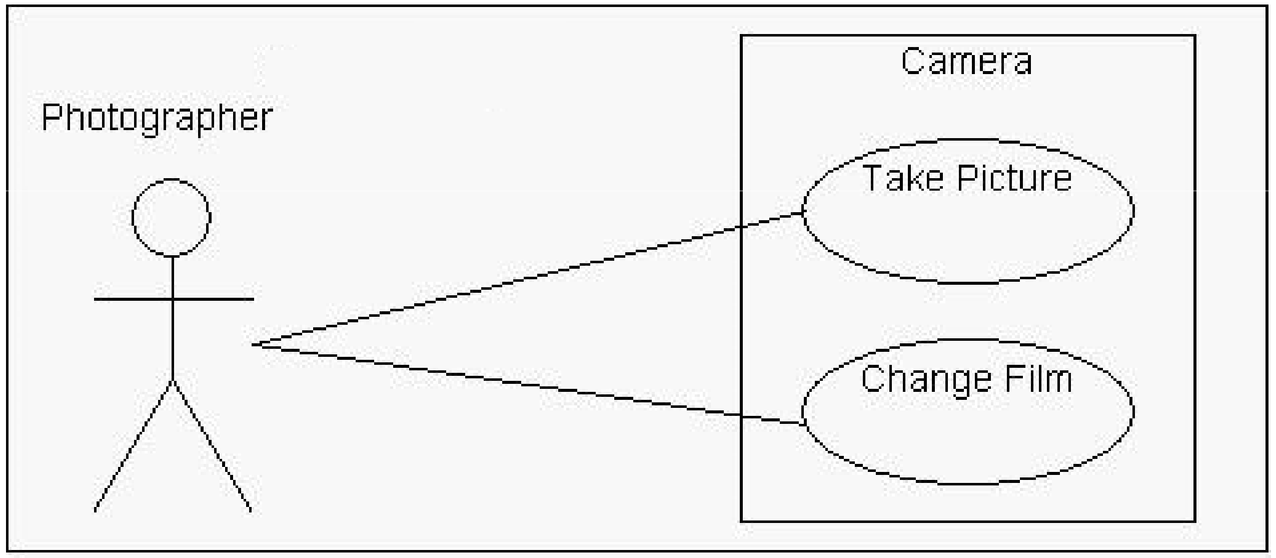


Image from: http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html

Requirements Workflow

Use Case: UML Diagrams

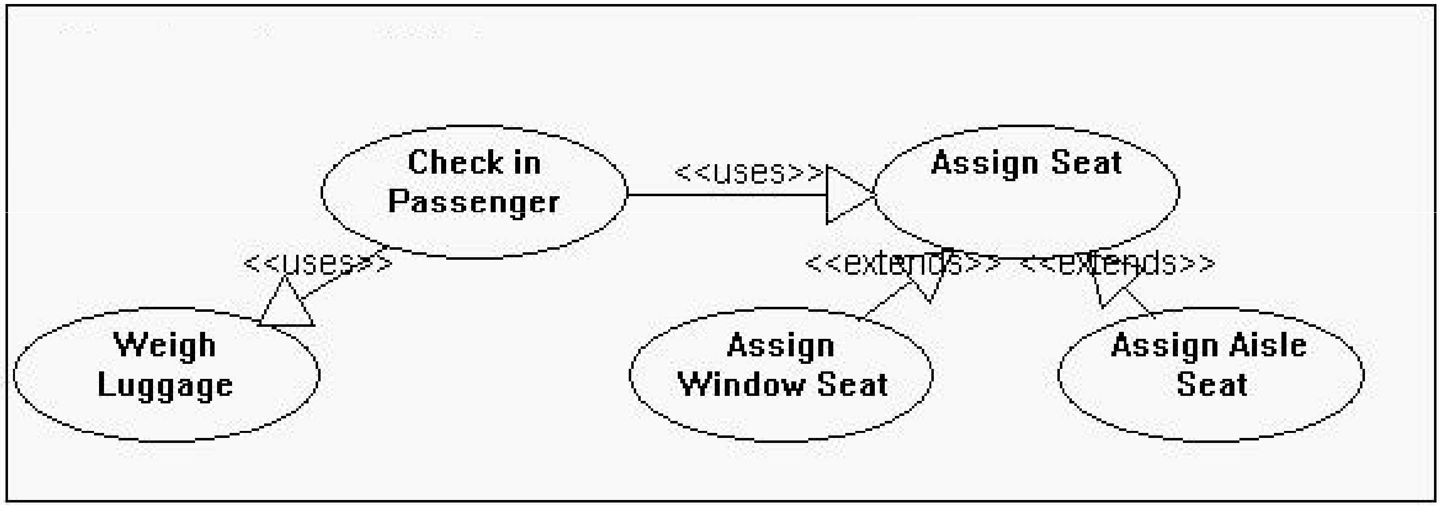


Image from: http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html

Requirements Workflow

Use Case: Some tips

- When you model a use case:
  - Names a single, identifiable, and reasonably atomic behavior of the system or part of the system.
  - Factors common behavior by pulling such behavior from other use cases that it includes.
  - Is described by a minimal set of scenarios that specify the normal and variant semantics of the use case.

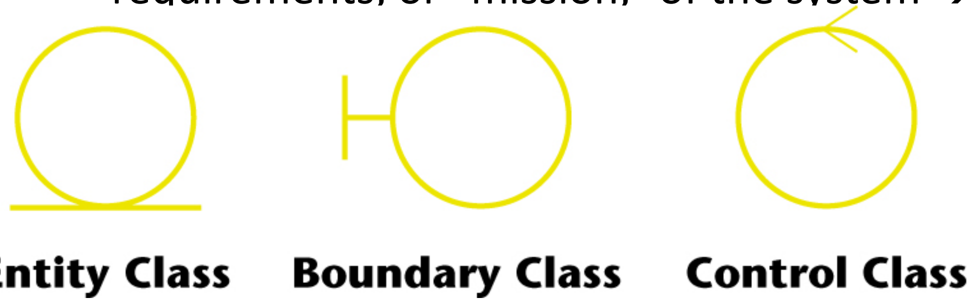
- When you draw a use case in the UML:
  - Show only those use cases that are important to understand the behavior of the system or the part of the system in its context.
  - Show only those actors that relate to these use cases.

Analysis Workflow

- To obtain a deeper understanding of the requirements
- To describe the requirements in a way that is
  - Easy to maintain, and
  - Provides insights into the structure of the target information system

Analysis Workflow

- To obtain a deeper understanding of the requirements
- To describe the requirements in a way that is
  - Easy to maintain, and
  - Provides insights into the **structure** of the target information system
    - What types of objects we're going to need to create and instantiate in order to represent the proper abstraction → **The static model**
    - How these objects will need to collaborate in carrying out the overall requirements, or "mission," of the system → **The dynamic model**



These are stereotypes (extensions of UML)

UML allows us to define additional constructs that are not part of UML but which we need in order to model a system accurately

Analysis Workflow

Identifying Appropriate Classes

- Noun Phrase Analysis
  - List
  - Filter
  - Groups

Remember, we're trying to identify both physical and conceptual objects

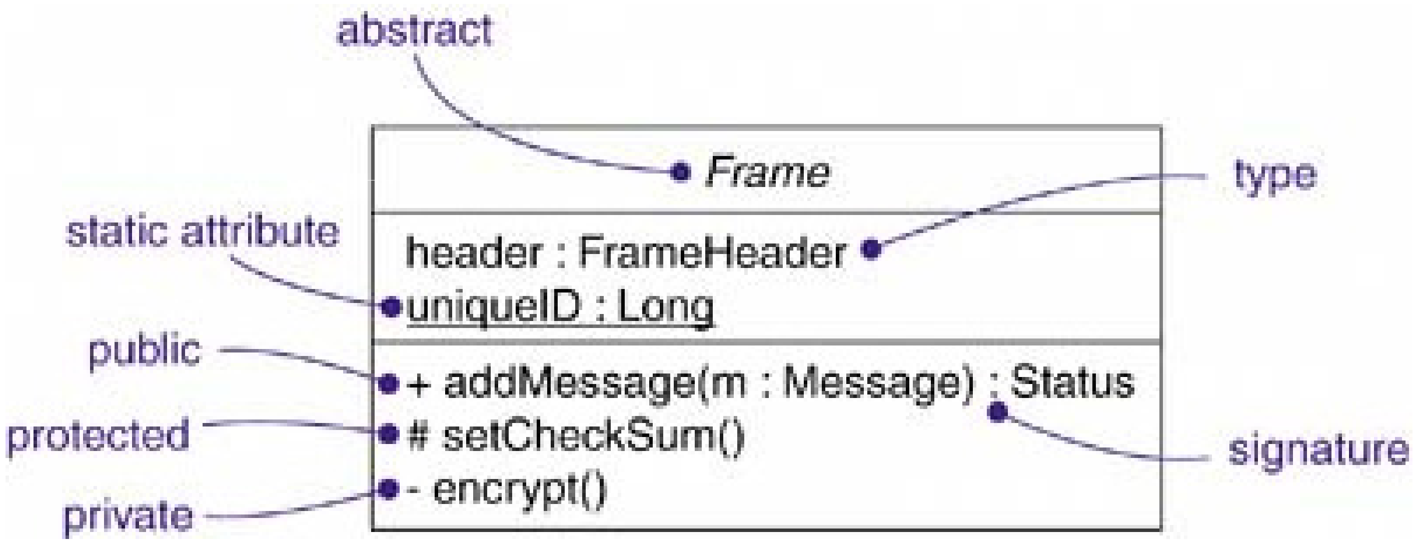
"something mental or physical toward which thought, feeling, or action is directed."

UML: Class Diagrams (Static modeling)

Sequence Diagrams (Dynamic modeling)

Analysis Workflow

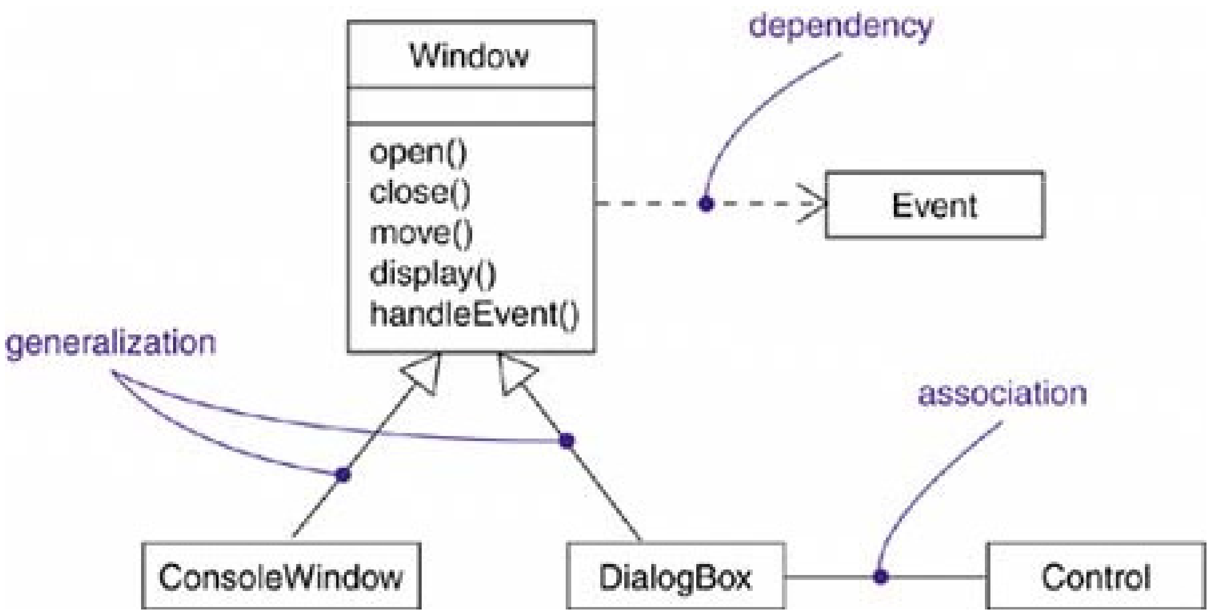
UML: Class Diagrams (Static modeling)



Analysis Workflow

UML: Class Diagrams (Static modeling)

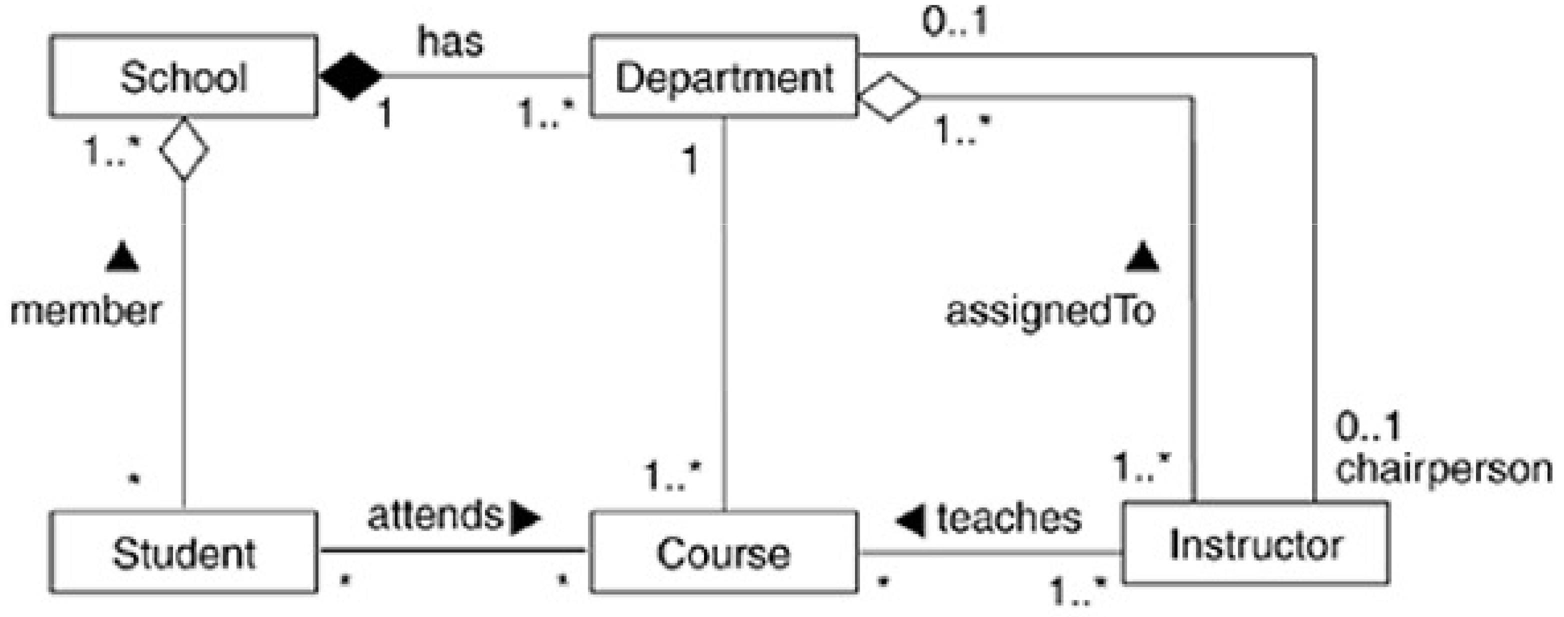
- Relationships Between Classes



Analysis Workflow

UML: Class Diagrams (Static modeling)

- Relationships Between Classes: Structural Relationships



Analysis Workflow

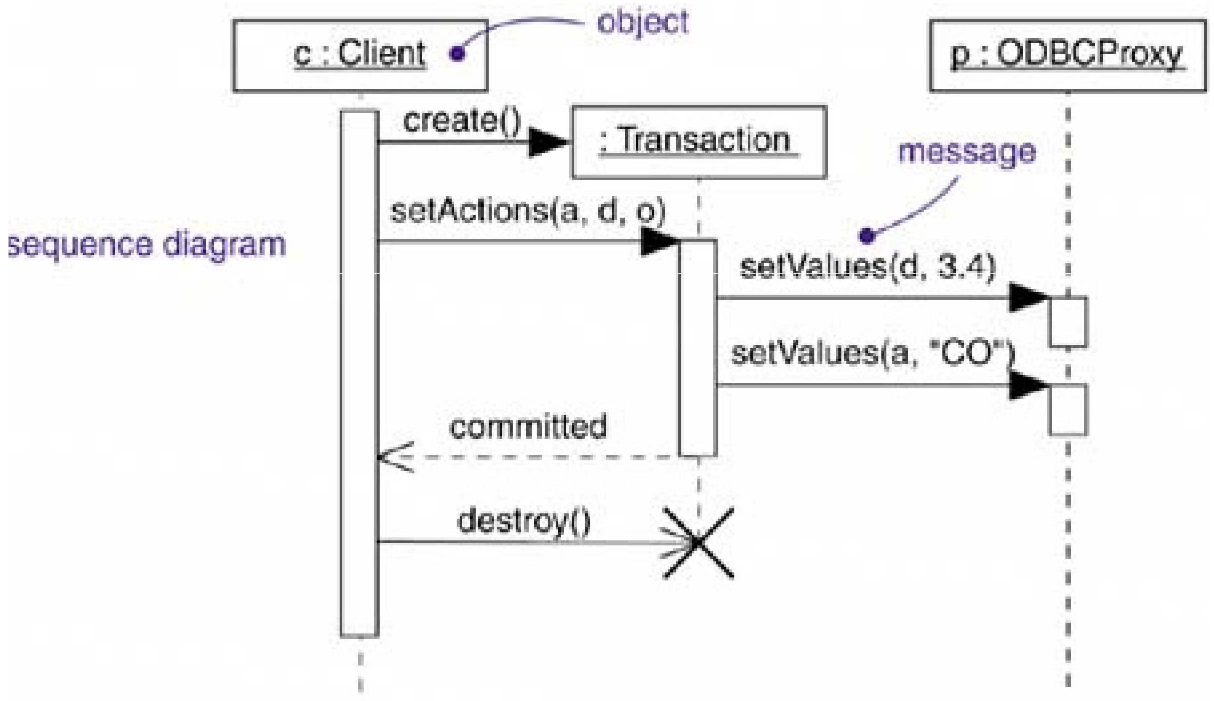
UML: Sequence Diagrams (Dynamic modeling)

- It is usually easy to extract boundary classes
  - Each input screen, output screen, and printed report is generally modeled by a boundary class

- It is also usually easy to extract control classes
  - Each nontrivial computation is generally modeled by a control class

Analysis Workflow

UML: Sequence Diagrams (Dynamic modeling)



References

1. J. Barker, *Beginning Java Objects: From Concepts To Code, Second Edition*, Apress, 2005.
2. S.R. Schach, Introduction to Object-Oriented Analysis & Design, McGraw-Hill/Irwin, 2004.
3. M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition), Addison-Wesley Professional, 2003.
4. G. Booch, J. Rumbaugh, and I. Jacobson, Unified Modeling Language User Guide, The (2nd Edition), Addison-Wesley Professional, 2005.