UML: An Overview
(Based on [Booch *et al.* 2005])

Yih-Kuen Tsay

Dept. of Information Management

National Taiwan University

Outline

- Introduction

- Basics of Modeling

- Overview of the UML

- Sample UML Diagrams

Introduction: History of the UML

- The UML---Unified Modeling Language, is a standard graphical language for "drawing a system's blueprints"

- It was initially the result of an effort in unifying the Booch, OOSE, and OMT methods

- Most major software companies eventually got involved, resulting in UML 1.1 (1997)

- Its maintenance was then taken over by OMG

- Adoption of a major revision---UML 2.0 was completed in 2005, also an ISO standard

- Most recent version: UML 2.5 (June 2015)

Intro.: What the UML Is For

- For "drawing a system's blueprints''

- More specifically, for

  - Visualizing
  - Specifying
  - Constructing
  - Documenting

  object-oriented, software-intensive systems.

  (This corresponds to the four aims of modeling.)

Intro.: Whom the UML Is For

- **Analysts and End Users**: specify the (structural and behavioral) requirements

- **Architects**: design systems that meet the requirements

- **Developers**: turn the design into executable code

- Others: quality assurance personnel (e.g., testers), technical writers, librarians, project managers, …

  All roles in software development should know something about the UML.

Importance of Modeling

- Mind the scale:
  - ❑ dog house
  - ❑ family house
  - ❑ office building
- The use of modeling is a common thread of successful software projects
- In fact, modeling can be found in every discipline/profession

Basics of Modeling

- **What** is a model?
  - ❑ simplification of reality
  - ❑ blueprints of a system: structural or behavioral
- **Why** do we model?
  - ❑ To better understand the system under development
  - ❑ To focus on one aspect at a time (it is not possible to comprehend a complex system in its entirety, so divide and conquer ...)

Four Aims of Modeling

- To visualize a system

- To specify its structure and/or behavior

- To provide a guiding template for construction

- To document the decisions made

More Tips

- Use a common language

- Do modeling now, before it is too late
  - Things may get more complex than expected

Principles of Modeling

- Models influence the solutions (so, choose your models well)

- Different levels of precision may be expressed

- Good models are connected to reality

- No single model is sufficient; multiple models/views are needed

Five Views of an Architecture

The four aims of modeling demand the system be viewed from different perspectives:

- Use case view: exposing the requirements

- Design view: capturing the vocabulary of the problem/solution space

- Interaction view: flow of control and messages

- Implementation view: physical realization

- Deployment view: system engineering issues

Object-Oriented Modeling

- The main building blocks of all software systems are objects and classes

- An object is a thing drawn from the vocabulary of the problem/solution space

- Every object has an identity, a number of states, and behavior

- A class defines a set of common objects

Overview of the UML

- Things
- Relationships
- Diagrams

The UML in the Software Development Process

- The UML allows one to express different views of a system and their interactions

- The UML is largely process-independent

- The OMG recommends using the UML with the so-called *Unified Software Development Process*:

  - Characteristics: (1) use case driven; (2) architecture-centric; (3) iterative and incremental (a series of executable releases; continuous integration)

  - Four phases: inception, elaboration, construction, transition

Things in the UML

- ## Structural Things
  - Class, interface, collaboration, use case, active class, component, artifact, node
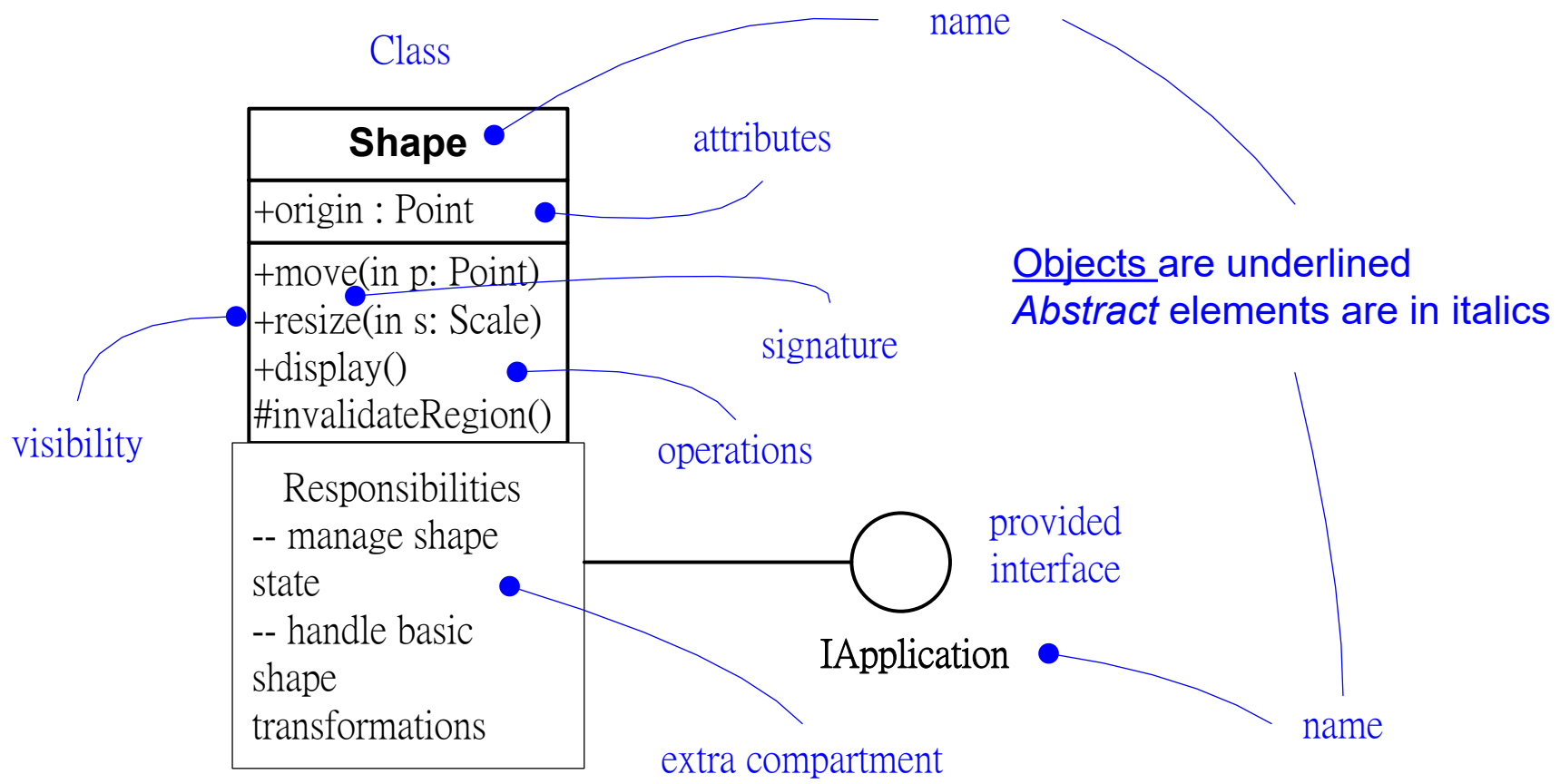
- ## Behavioral Things
  - Interaction (messages, action sequences, links)
  - State machine (states, transitions, events)

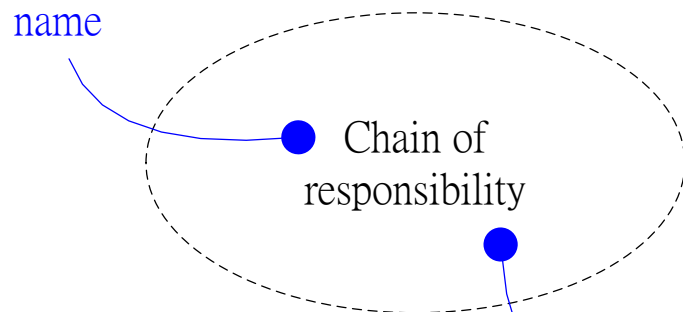- ## Grouping Things: packages

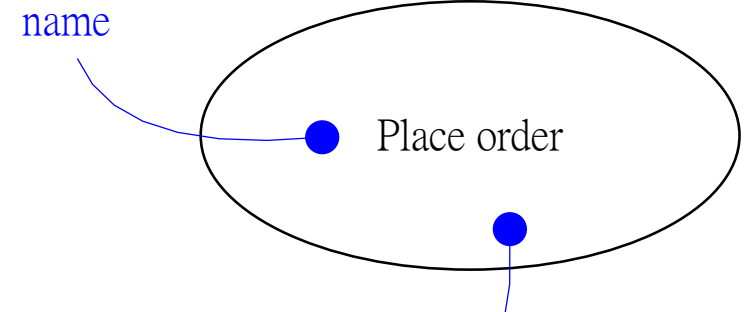- ## Annotational Things

Structural Things (I)

Class

Shape

+origin : Point

+move(in p: Point)
+resize(in s: Scale)
+display()
#invalidateRegion()

Responsibilities
-- manage shape
state
-- handle basic
shape
transformations

name

attributes

Objects are underlined
Abstract elements are in italics

signature

operations

visibility

provided
interface

IApplication

name

extra compartment

Structural Things (II)

Collaboration

Use Case

name

name

Chain of responsibility

Place order

Extra compartments may be used to show contents.

Structural Things (III)

Active Class

Component

Node

name

attributes

name

icon

name

EventManager

-q : EventQueue

+suspend()
+flush()

Server

SpellChecker

<<manifest>>

<<artifact>>
spelling.dll

operations

Extra compartments may be used to
show contents.

Artifact

Behavioral Things (I)

Interaction

role
name
type
anonymous role

t:Thread

: Toolkit

sequence
label

a1 : run(3)

lifeline

run()

callbackLoop()

message

call

creation

<<create>>

p:Peer

execution
specification

handleExpose()

recursion

return

<<destory>>

destructon

Behavioral Things (II)

State Machine

final state

initial state

state

transition

nested state

guard

off

onHook

Working

ready(3) [signalOK]

Idle

keepAlive / check()

Connecting

Connected

Internal transition

offHook / reclaimConnection()

event

action

Grouping Things

Package

name

Business rules

Extra compartments may be used to show contents.

Annotational Things

Note

note

Consider the use
of the broker design
pattern here.
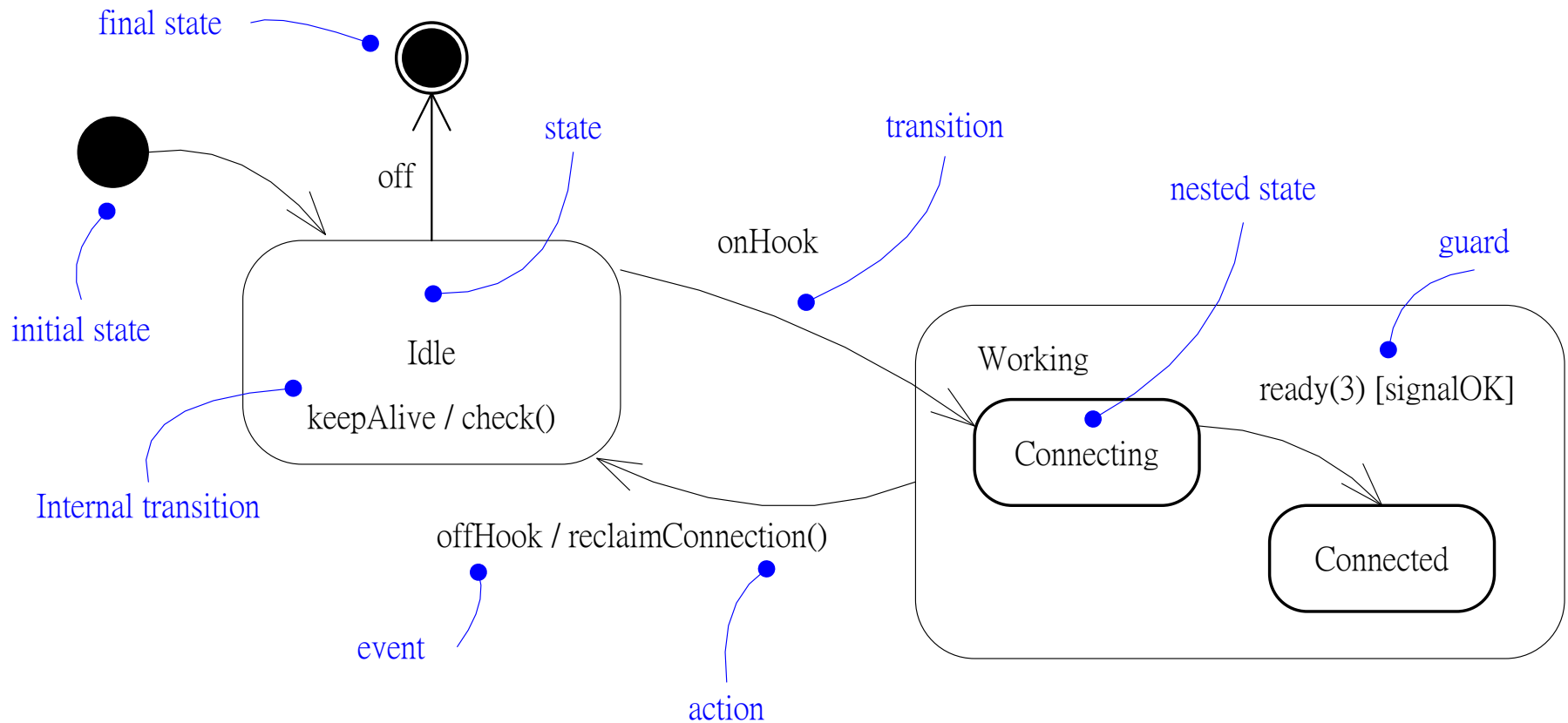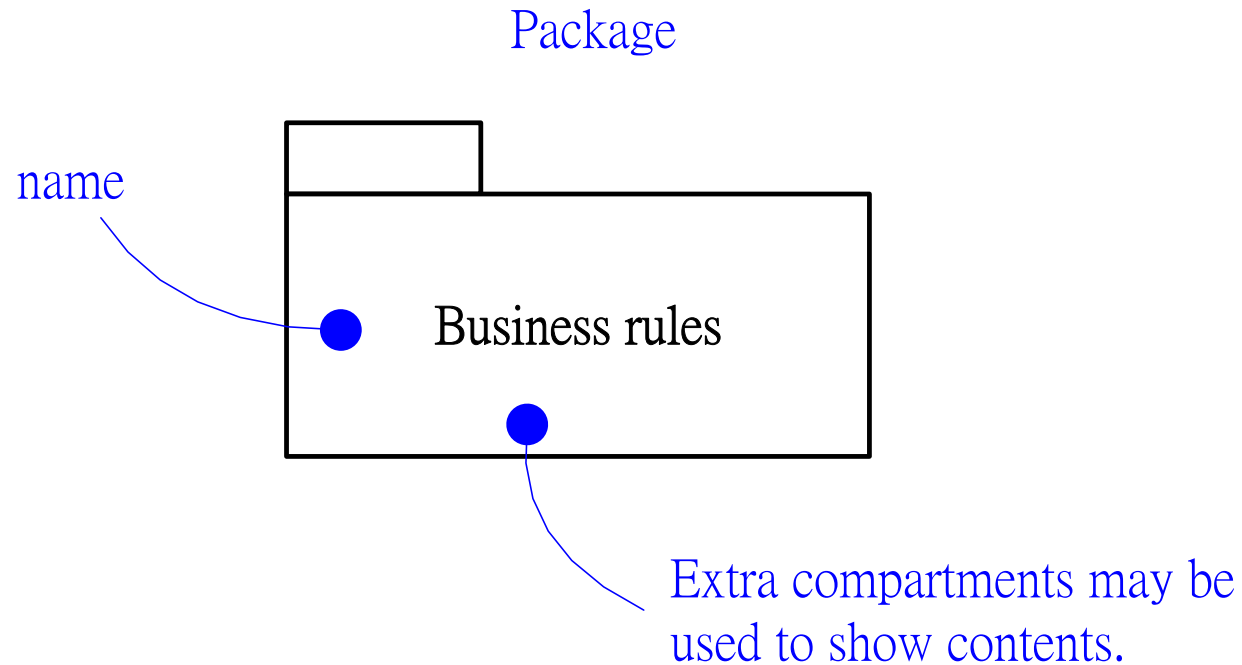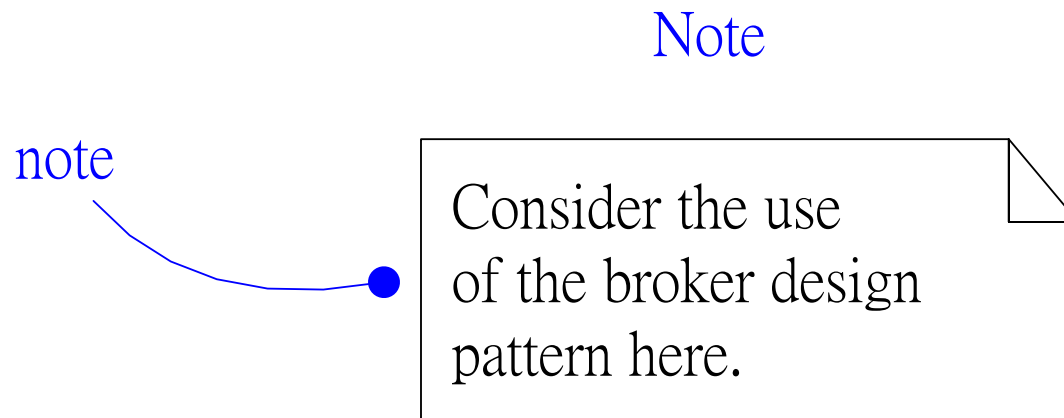
Relationships in the UML

- Dependency

- Association

- Generalization

- Realization

Relationships

name

Dependency   source   ● Owner   ----------→ target

name   multiplicity   navigation

Association   end   0..1   ● Employs   ● * ●   end

employer   employee

end name

(subclass)   (superclass)

Generalization   child ————————▷ parent
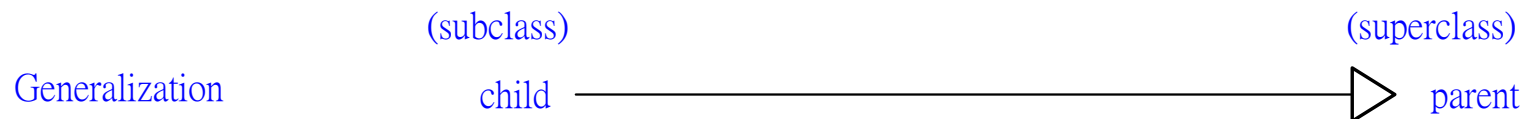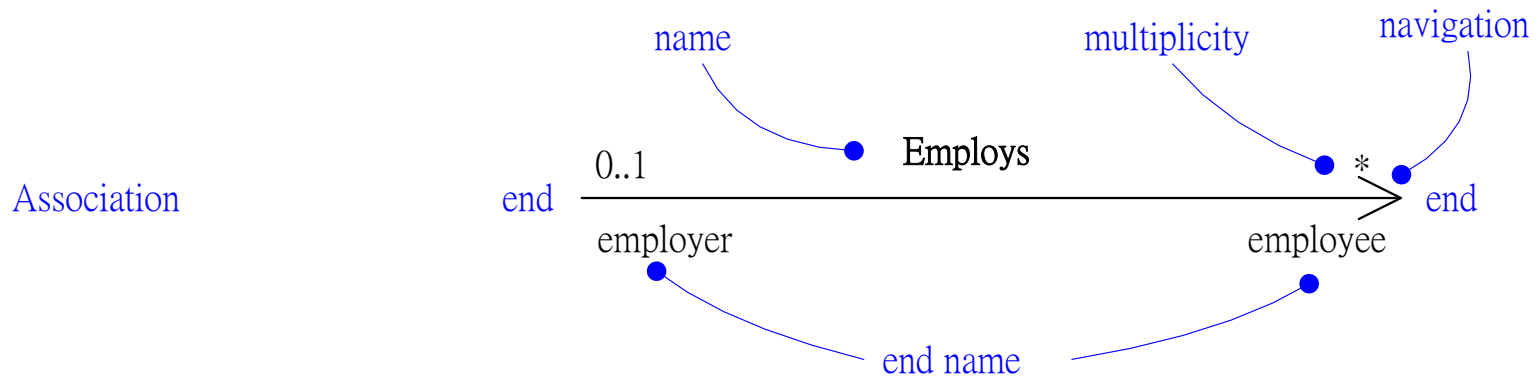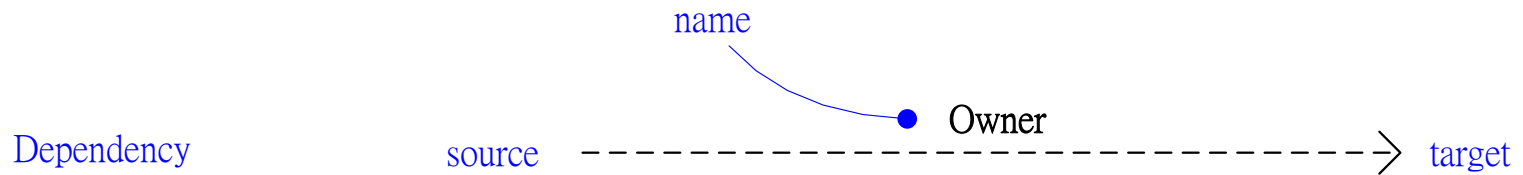
Note: direction of an association should now be indicated by a solid triangle ▸ following the association name.

Diagrams in the UML

- Graphical representations of things and relationships

- Structural and Architectural Diagrams:
  - class diagrams, object diagrams, component diagrams, composite structure diagrams, deployment diagrams (including artifact diagrams), package diagrams

- Behavioral Diagrams:
  - use case diagrams, interaction (sequence and communication) diagrams, state diagrams, activity diagrams, timing diagrams, interaction overview diagrams

Rules of the UML
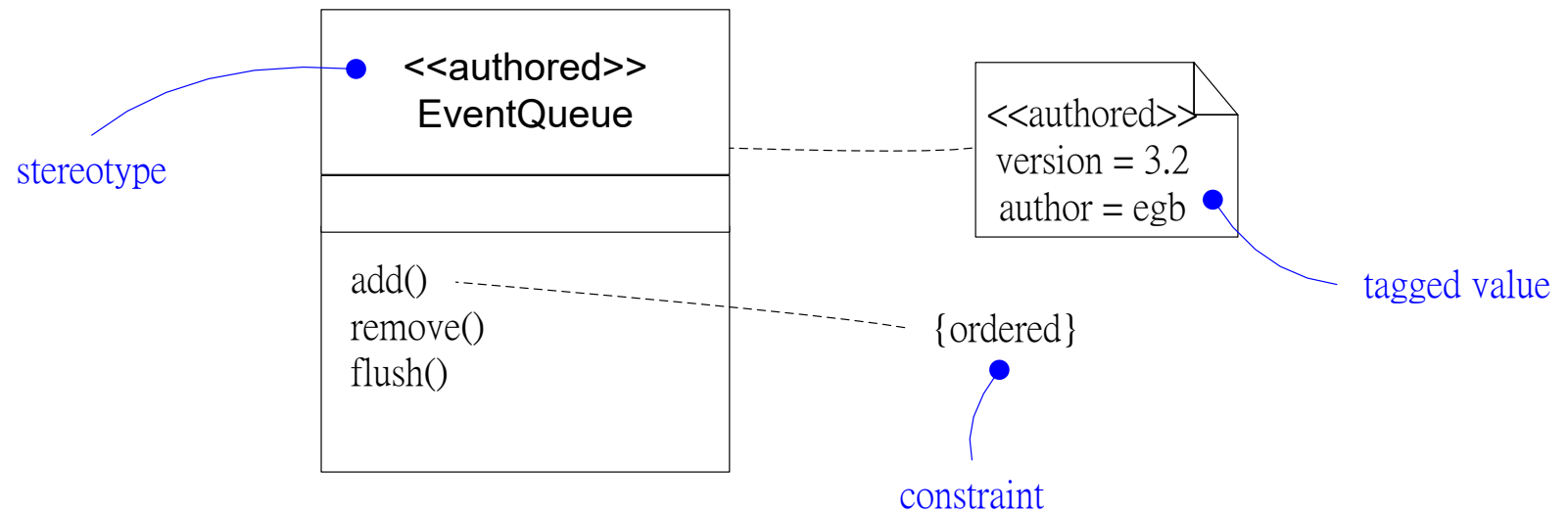
- ## Well-formed models
  - ❑ Self-consistent
  - ❑ Following UML rules for
    names, scope, visibility, integrity, execution
- ## Not well-formed models
  - ❑ Elided: some elements hidden
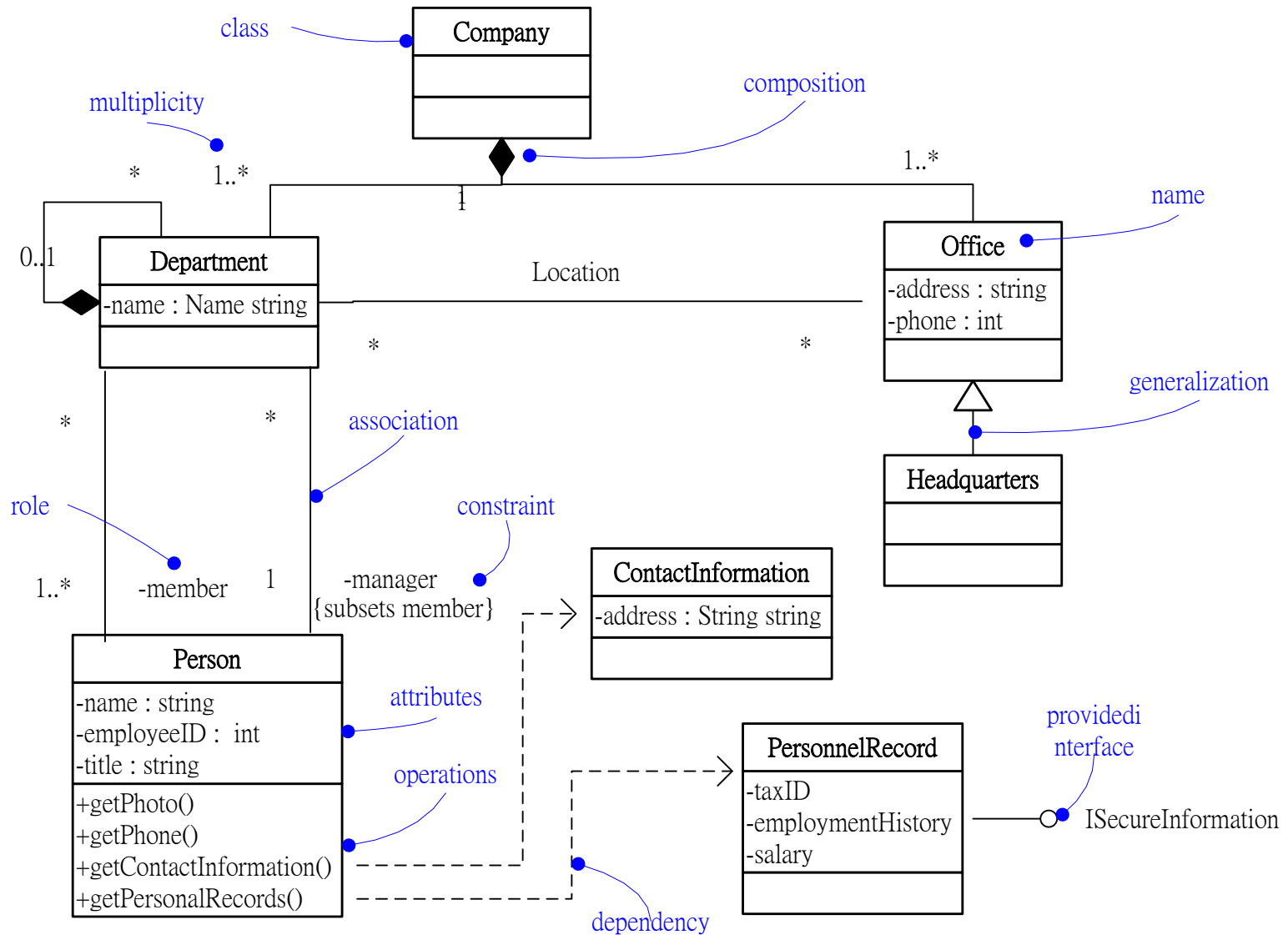 - ❑ Incomplete: some elements missing
  - ❑ Inconsistent

Common Mechanisms in the UML

- Specifications: textual statements behind every graphical element

- Adornments
  - ❑ unique notations for different elements/details

- Common divisions
  - ❑ class vs. object, interface vs. implementation, role vs. type

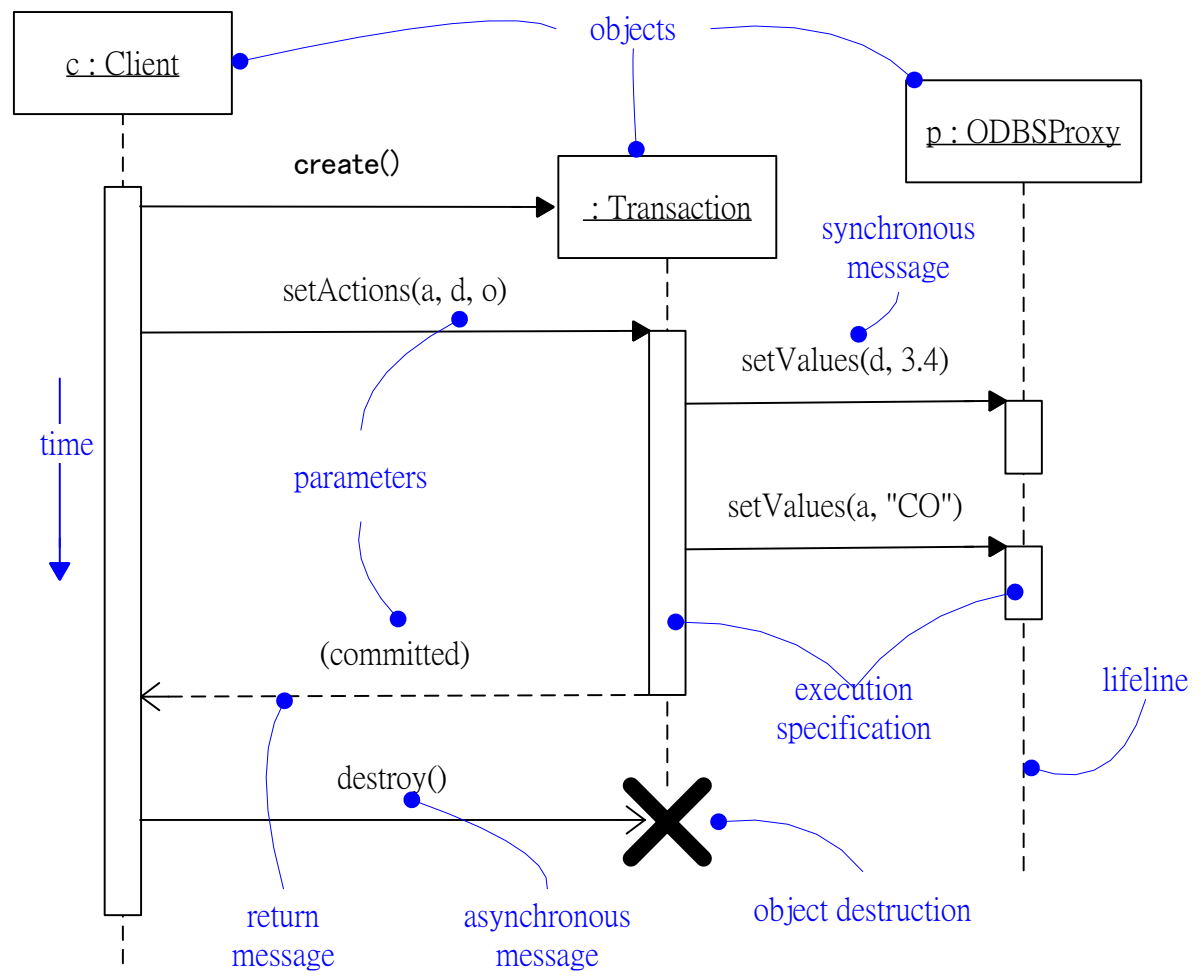- Extensibility mechanisms
  - ❑ stereotypes, tagged values, constraints

Extensibility

stereotype

<<authored>>
EventQueue

add()
remove()
flush()

<<authored>>
version = 3.2
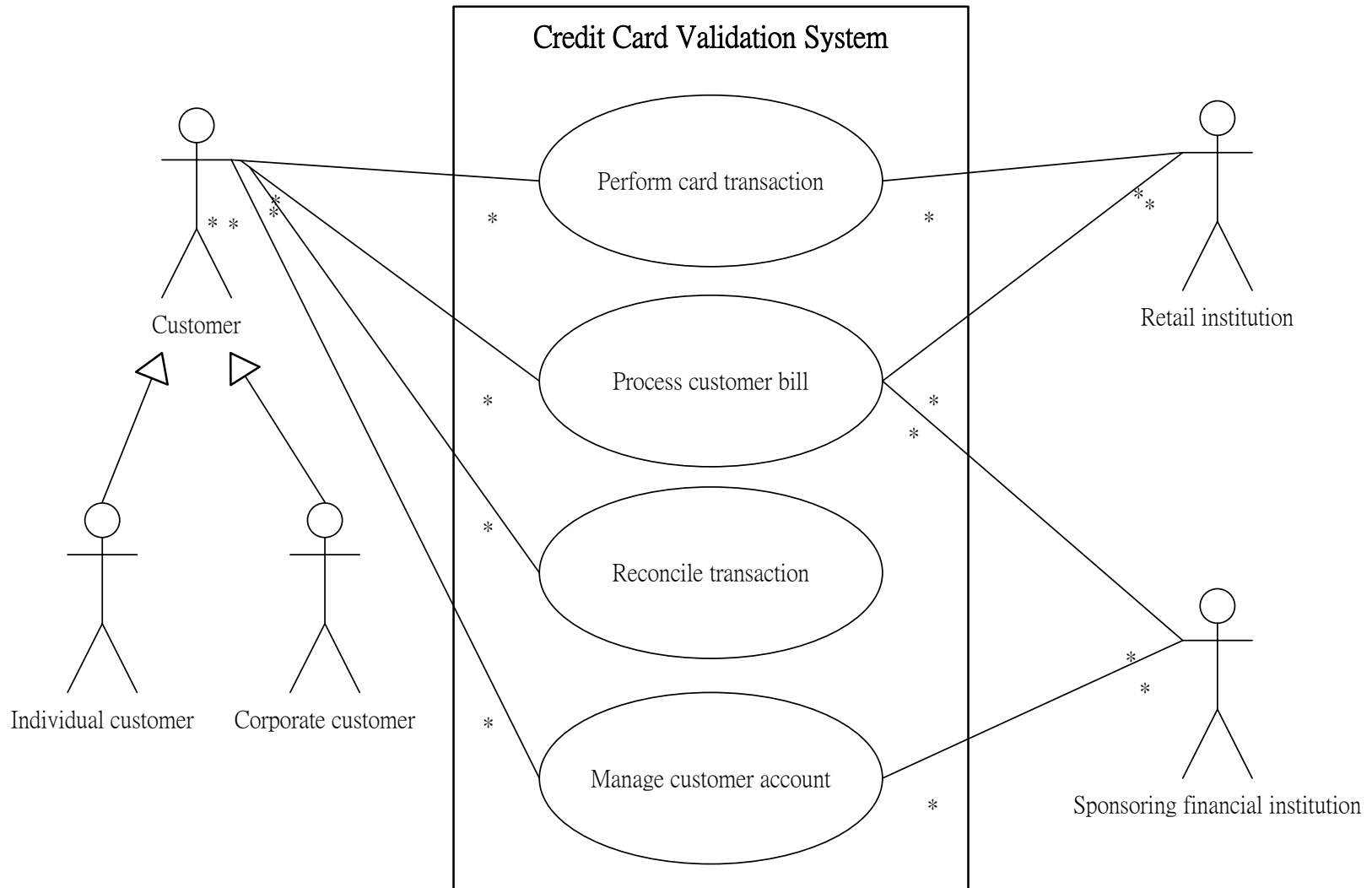author = egb

tagged value

{ordered}

constraint

Class Diagram

Sequence Diagram

Use Case Diagram

Remarks

- The best way to learn the UML is by actually using it:

  - Data modeling

  - Design patterns

  - Term project

- In follow-up lectures, we will cover

  - Basic structural and behavioral diagrams

  - Some more advanced UML features

  - The Object Constraint Language

- Things not covered in class are left for you to explore.